Jari Suninen

# Servomoottorin ohjaus Beckhoff Twincat 3:lla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

28.5.2018

Metropolia

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Jari Suninen<br>Servomoottorin ohjaus Beckhoff Twincat 3:lla<br><br>64 sivua + 9 liitettä<br>28.5.2018 |
| Tutkinto | Insinööri (AMK) |
| Tutkinto-ohjelma | Sähkö- ja automaatiotekniikka |
| Ammatillinen pääaine | Automaatiotekniikka |
| Ohjaajat | Lehtori Timo Tuominen |

Työn pääasiallisena tavoitteena oli toteuttaa Metropolia Ammattikorkeakoulun automaatiolaboratorioon servomoottorikäytön laitteisto, jolla pystytään seuraamaan käytön prosessin aikaisia muuttujia. Näitä ovat mm. moottorin nopeuteen, kiihtyvyyteen, paikkaan ja momenttiin liittyviä arvoja. Määrittelyn mukaisesti tavoitteena oli luoda moottorikäytölle paikallinen graafinen käyttöliittymä (GUI), jonka kautta prosessia voi ohjata ja valvoa muuttujia.
Toissijaisena tavoitteena oli tutkia Twincat-ADS viestintäprotokollan rakennetta ja selvittää kuinka tätä protokollaa käyttäen voisi luoda etäpalvelin sovelluksen. Etäpalvelinsovellus olisi eräänlainen ”soft-PLC”:hen perustuva web-käyttöliittymä, jolla voisi ohjata ja valvoa käyttöä IP-verkon kautta.

Käytön PLC-ohjelman ja käyttöliittymän rakentaminen toteutettiin Twincat 3 engineering (XAE) kehitysympäristön tarjoamilla työkaluilla Beckhoff CP6223 teollisuus-PC:ssä. Projekti koottiin järjestelmähallinnan hakemistopuuhun, johon luotiin PLC-ohjelma, käyttöliittymä, sekä yhdistettiin I/O-väylään ja toimilaitteisiin. Käytetyn I/O-väylän, Ethercatin väylä-pääteaste EK1100 oli yhteydessä PC:n Ethernet-kaapelin välityksellä. I/O-väylään kytkettiin moottorin servovahvistin, johon puolestaan liitettiin itse moottori ja takaisinkytketty, paikan ilmaiseva pulssianturi.

Moottorikäyttöön luotiin PLC-ohjelma, joka käyttöliittymän kautta käyttää moottoria "PLCopen" standardin liiketilojen mukaisesti. Liiketilat olivat paikka-ohjattu, erillisen liikkeen liiketila sekä jatkuvan liikkeen liiketila. PLC-ohjelmassa näitä liiketiloja vastaavat funktioblokit olivat "MC_MoveAbsolute" sekä "MC_Jog". Käyttöliittymän painikkeet korreloivat näiden funktioblokkien operointia. Prosessin muuttujien tilaa käsittelevää informaatiota esitettiin käyttöliittymästä numeeristen tekstikenttien välityksellä, josta näkyy moottorin nopeus (mm/s) sekä kuljettu matka (mm) suhteessa nollattuun referenssipisteeseen.

Työn päätavoite saavutettiin. Servomoottorikäytön rakentaminen Twincat 3-ympäristöön, jossa on paikallinen käyttöliittymä. Palvelinsovelluksen luominen ADS-protokollan mukaiseen viestintään ”. NET” C#-kielellä osoittautui liian laajaksi ja aikaa vieväksi tämän projektin osalta. Itse C#-kielen opiskelua voidaan pitää tämän kohdalla saavutettuna tavoitteena.

| | |
|---|---|
| Avainsanat | Liikkeen-ohjaus, PLC,GUI |

| | |
|---|---|
| Author<br>Title<br><br>Number of Pages<br>Date | Jari Suninen<br>Control of Servo Drive with Twincat 3<br><br>64 pages + 9 appendices<br>28 May 2018 |
| Degree | Bachelor of Engineering |
| Degree Programme | Electrical and automation engineering |
| Professional Major | Automation Technology |
| Instructors | Timo Tuominen, Senior Lecturer |

The purpose of this final year project was to build a demonstration servo motor drive platform for the Metropolia University of Applied Sciences automation laboratory. The primary goal was to implement the servo drive environment to the Beckhoff Twincat 3 PLC platform. Main requirements to this project were to create a motion control PLC program and local graphical user interface (GUI) application to operate the drive. User interface would also have to display essential motion control process values to the operator, like motor velocity and position. Secondary goal was to examine the Twincat ADS communication protocol and whether it would be feasible using it to create a server interface (HMI-Human Machine Interface) application to control and monitor the drive through IP network.

The creation of this project was done on tools supplied by Twincat 3 engineering (XAE) development environment. Twincat 3 software was installed on Beckhoff CP6223 industrial panel PC. Motion control project was created to the system manager directory tree on which the whole project was built upon and to which all the modules of the servo drive were added. The main configuration sections on this directory tree were system, motion control, PLC and I/O. In motion control section drive's numerical control axis (NC) was created and added to the project. Linking of the I/O to NC and linking of NC to PLC program also happened there. I/O section was all about engine parameters. Finally the creation of PLC program and local graphical interface (GUI).

The end result of the project was a basic PLC program that had motion profiles that were in compliance with "PLCOpen" standard of motion states and their corresponding function blocks (FB). Discrete FB's were "MC_MoveAbsolute" for position feedback control movements ja "MC_Jog" for continuous manual movements. The user interface (GUI) had control buttons to activate the drive and select these motion states. It also had text box objects for reading position parameters, as well as text field objects for displaying velocity and travelled distance values.

In conclusion the main objective was achieved. Operating servo environment with local GUI. As for the examination of ADS protocol and creating a server application in.NET C# language, it was too time consuming for this project but served well for studying C#.

| | |
|---|---|
| Keywords | PLC, Motion Control,GUI |

## Sisällys

## Lyhenteet

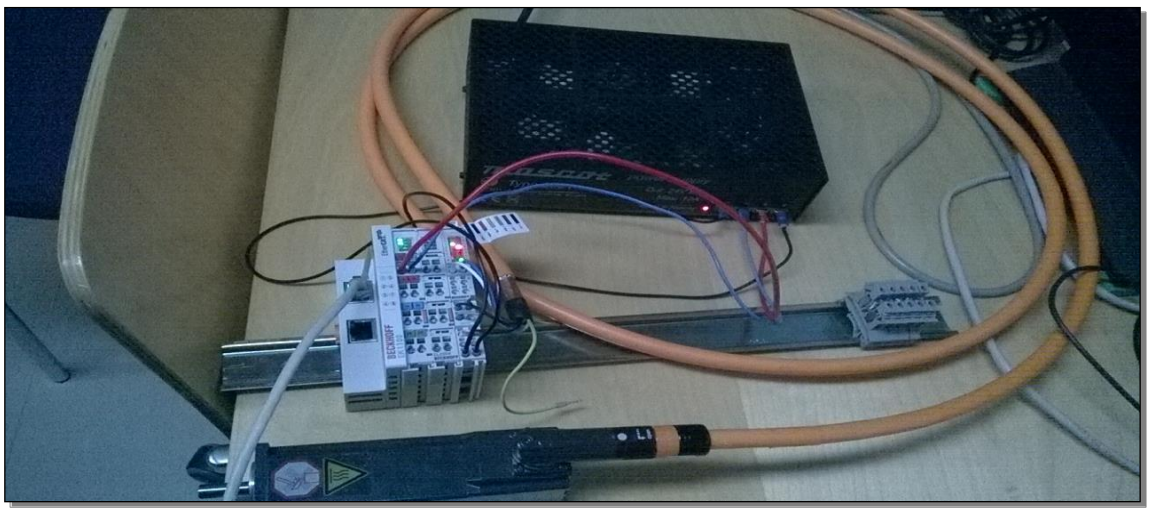
| | |
|---|---|
| GUI | Graphical User Interface. Graafinen käyttöliittymä. |
| PLC | Programmable Logic Controller. Ohjelmoitava logiikka. |
| PLC RT | PLC Real Time. Reaaliaikaisen PLC-rutiinin palvelin. |
| HMI | Human Machine Interface. Valvomo-ohjelmisto. Prosessin ohjauksen rajapinta operaattorin ja prosessin välissä. |
| NC | Numerical Controller. Liikkeen ohjain. |
| I/O | Input/Output.Tulomuuttujat/lähtömuuttujat. |

# 1 Johdanto

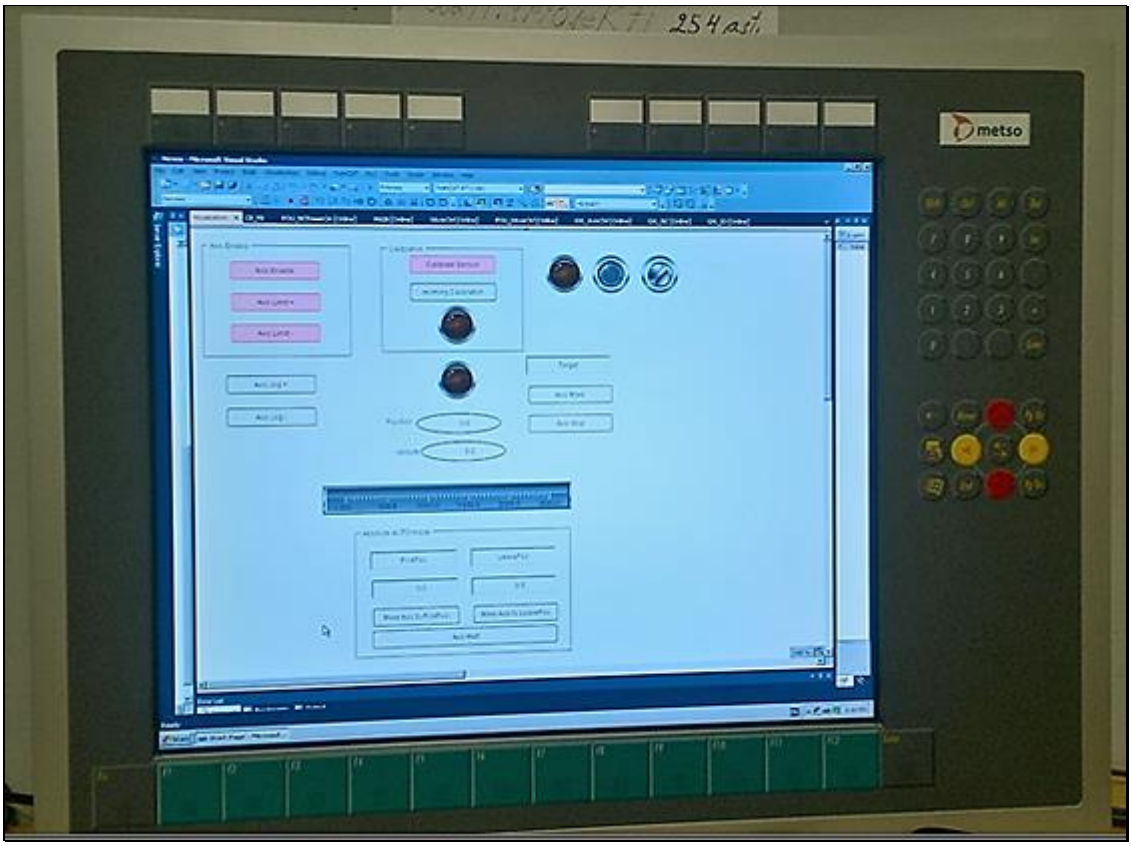
Työn tarkoituksena oli rakentaa Metropolian Myyrmäen yksikön automaatiolaboratorioon moottorikäytön demonstraatiolaitteisto Beckhoffin Twincat 3 PLC järjestelmän ympärille.

Tavoitteena oli toteuttaa laitteisto, jolla sähkömoottorikäytön ohjauksen lisäksi pystytään seuraamaan prosessin aikaisia muuttujia. Nämä ovat mm. moottorin nopeuteen, kiihtyvyyteen, paikkaan liittyviä arvoja sekä sähköteknisiä suureita, kuten jännite ja virta. Määrittelyn mukaisesti tavoitteena oli luoda moottorikäytölle graafinen käyttöliittymä (GUI), jonka kautta prosessia voi ohjata ja valvoa muuttujia. Projektin toteutuksessa ja käyttöliittymän rakentamisessa käytettiin Twincat 3 (XAE) -kehitysympäristöä ja C#-ohjelmointikielen kirjastoja.

Ympäristön laitteisto koostui näytön ympärille integroidusta ”Beckhoff CP6223" -paneeli-IPC:stä, johon on asennettu Twincat 3 Engineering-ohjelmisto., 6-napaisesta kestomagnetoidusta DC-moottorista(AM8113-0F20-0000), servovahvistimesta(EL7201) ja Ethercat-väylän pääteasteesta(EK1100) paneeli-PC:n ja servokäytön välillä. Näiden lisäksi kokonaisuuteen kuului virtalähteet paneeli-PC:lle ja servokäytölle (MASCOTT), sekä erikoiskaapelointi moottorin ja servokäytön välillä. Kuvasta 1. ja 2. näkyy laitteistokokonaisuus.



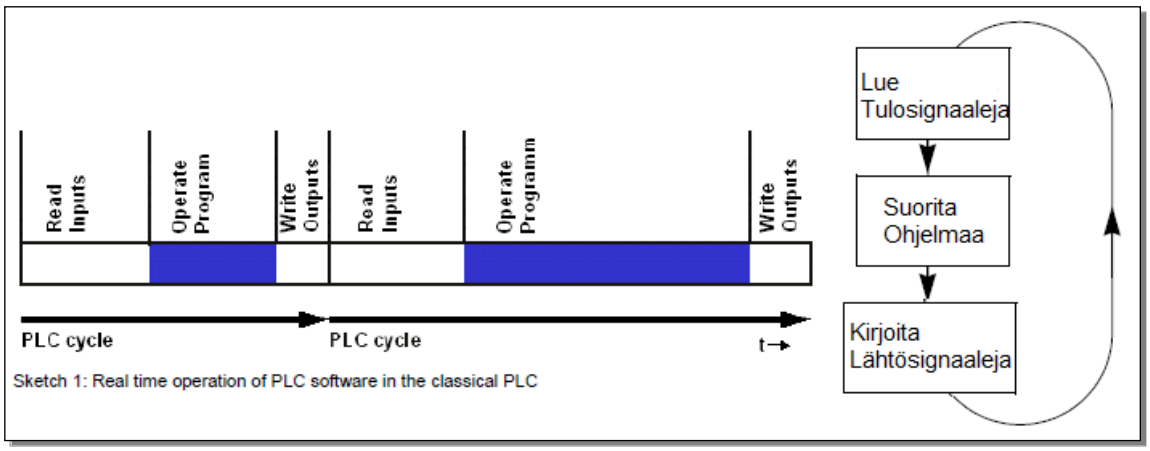
Kuva 1. Kuva servomoottorista ja I/O-yksiköstä.




Kuva 2. Työn paneeli-PC, jossa on integroitu keskusyksikkö ja näyttö.

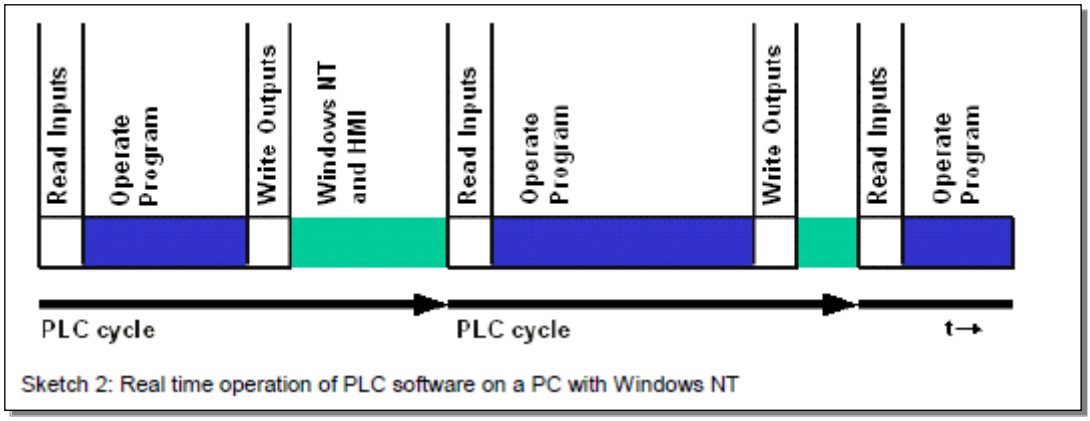
## 2 Yleistä Twincat-järjestelmästä ja soft-PLC:stä

Reaaliaikaisen, klassisen PLC:n operaatioiden suoritus sykli näkyy kuvassa 3. Rutiine- ja on kolmea lajia: lukurutiinit, PLC-ohjelman suorituksen rutiinit ja lähtösignaaleiden kirjoitusrutiinit.



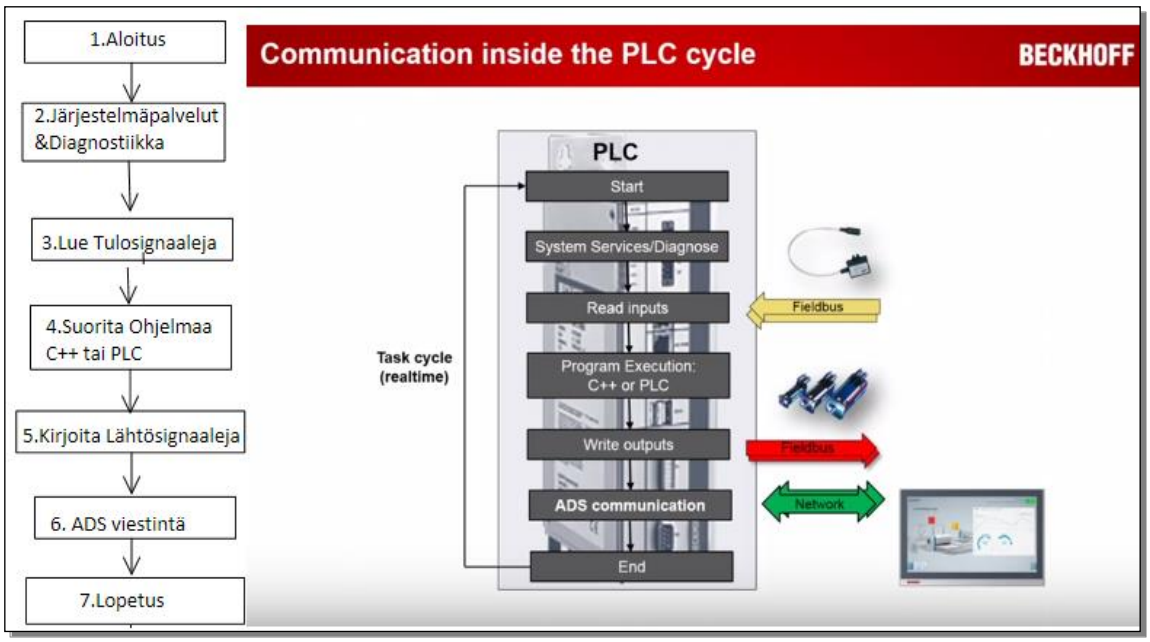

Kuva 3. Klassisen PLC:n suoritussykli, jossa syklit ovat erimittaisia.

Kuvassa 4. on nykyaikaisen, PC-koneen tai muun tietokoneen käyttöjärjestelmässä toimivan ohjelma-PLC:n reaaliaikainen suorituskaavio. Rutiineihin on lisätty aikaviipaleet käyttöjärjestelmän ja esim. käyttöliittymän (HMI) operaatioille. Lisääntyneet rutiinit on jaettu tasamittaisiin sykleihin, jolloin resursseja ja kapasiteettia voidaan jakaa järkevästi.




Kuva 4. Ohjelmallisen PLC:n suoritussykli on jaettu tasaisin aikajaksoihin.

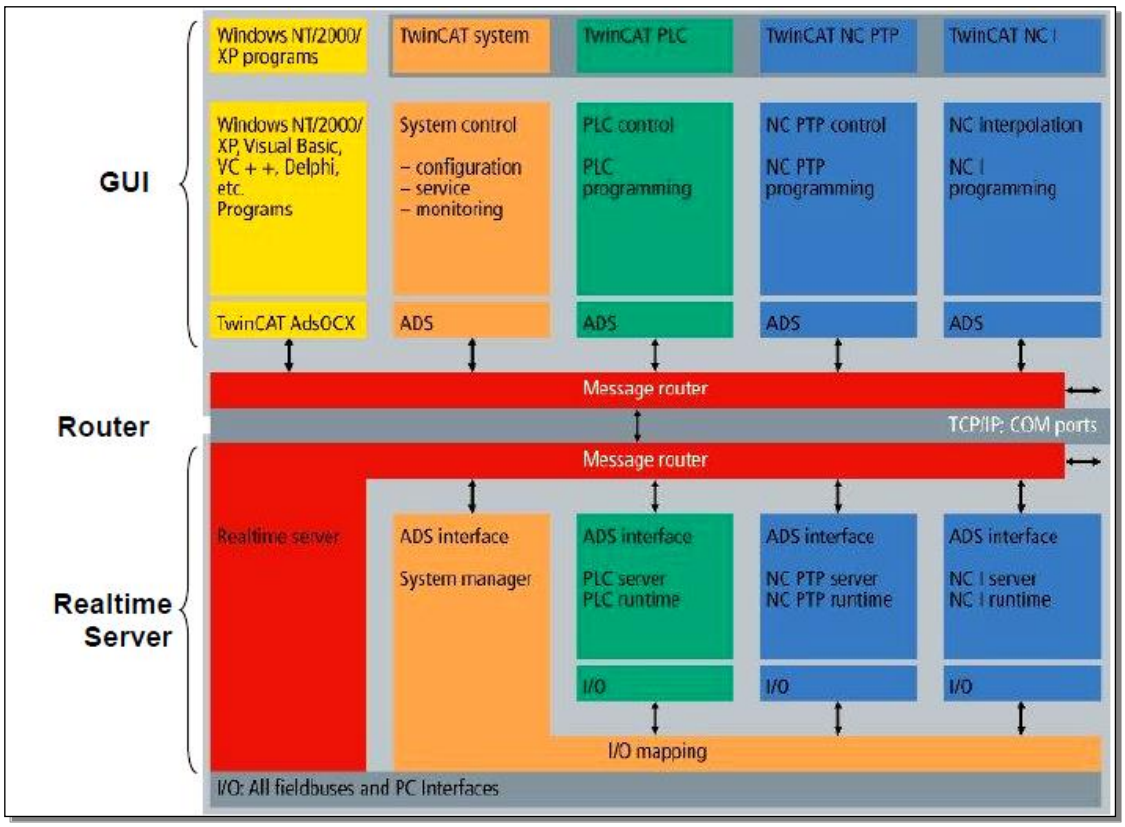
Kuvassa 5. on Twincat 3:n reaaliaikainen PLC suoritussykli. Yhdessä syklissä on jo seitsemän rutiinia.




Kuva 5. Twincat 3:n suoritussykli.[3]

## 3 Twincat 3-järjestelmäarkkitehtuuri

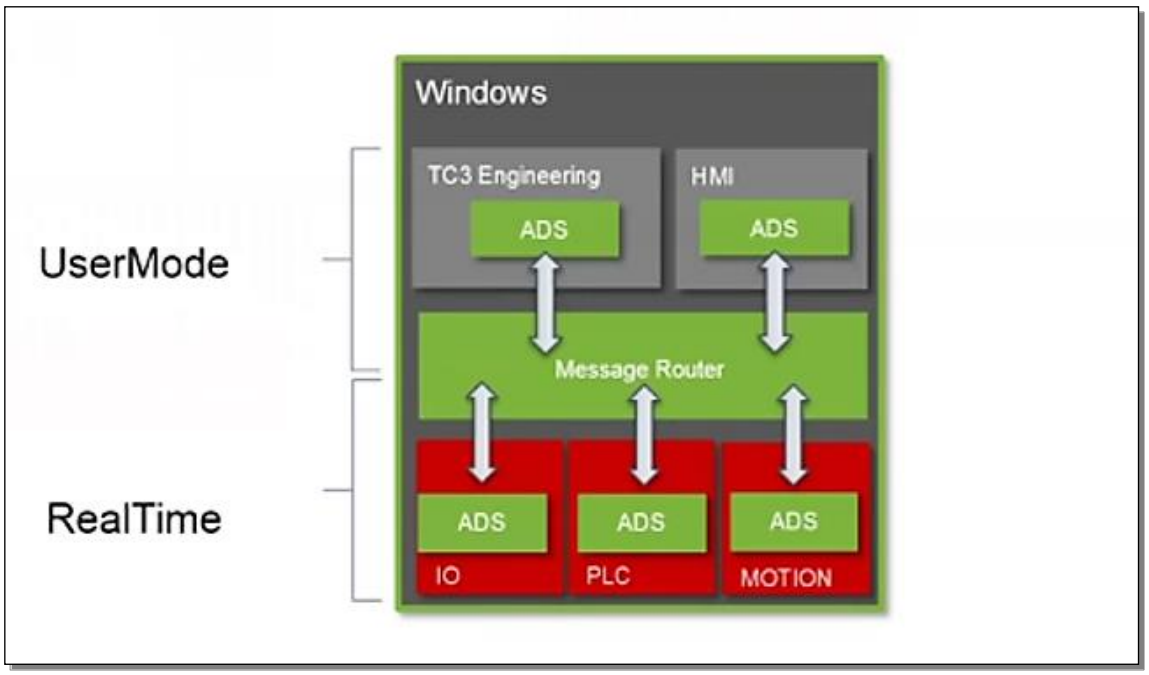
Twincatin järjestelmäarkkitehtuurissa jokaisella ohjelman osa-alueella on oma itsenäinen rooli. Itsenäisiä ohjelma moduuleita voidaan ajatella fyysisenä laitteena ja niitä on jaoteltu jokaiseen käyttötarkoitukseen. Nämä ohjelma moduulit toimivat palvelin- tai asiakas-periaatteella vastaten käyttäytymiseltään fyysistä laitetta. Näin niitä voidaan kutsua virtuaalisiksi laitteiksi. Asiakas-ohjelmat pyytävät palveluita palvelin-ohjelmilta. Automaatio sovelluksissa, kuten moottorikäytöissä ja liikkeen ohjauksessa, tärkeimmät virtuaaliset laitteet ovat kuvissa 6. ja 7. näkyvät PLC logiikka ohjauksen moduuli (”**PLC** System”), numeerisen liikkeenohjauksen moduuli (”MOTION/**NC** System”) sekä I/O-moduuli. [6]




Kuva 6. Twincatin järjestelmäarkkitehtuuri.[6]

Yksinkertaistettuna Twincat 3:n laitekonsepti on nähtävissä kuvassa 7. Toiminta jaetaan käyttäjätilaan ”User mode” ja reaaliaikaiseen tilaan ”Real Time”.

Ylhäällä kuvassa näkyy käyttäjätilan ohjelman kehitysympäristö (TC3 Engineering), jossa tapahtuu kaikki ohjelman kehitys ja ohjelmointi. Tämä koskee niin reaaliaikaiseen PLC puolen ohjelmia kuin esim. korkeamman tason kielillä toteutettujen sovellusten ohjelmia. Ylhäällä oikealla näkyy myös paikallisen käyttöliittymän ohjelmamoduuli (HMI). Kuvan alareunan punaiset laatikot ovat virtuaalilaitteet/ohjelmamoduulit I/O, PLC sekä MOTION/NC. Keskellä kuvaa on viestin reititin (”Message Router”), johon kaikki ohjelmamoduulit ovat yhteydessä ADS-rajapinnan kautta.



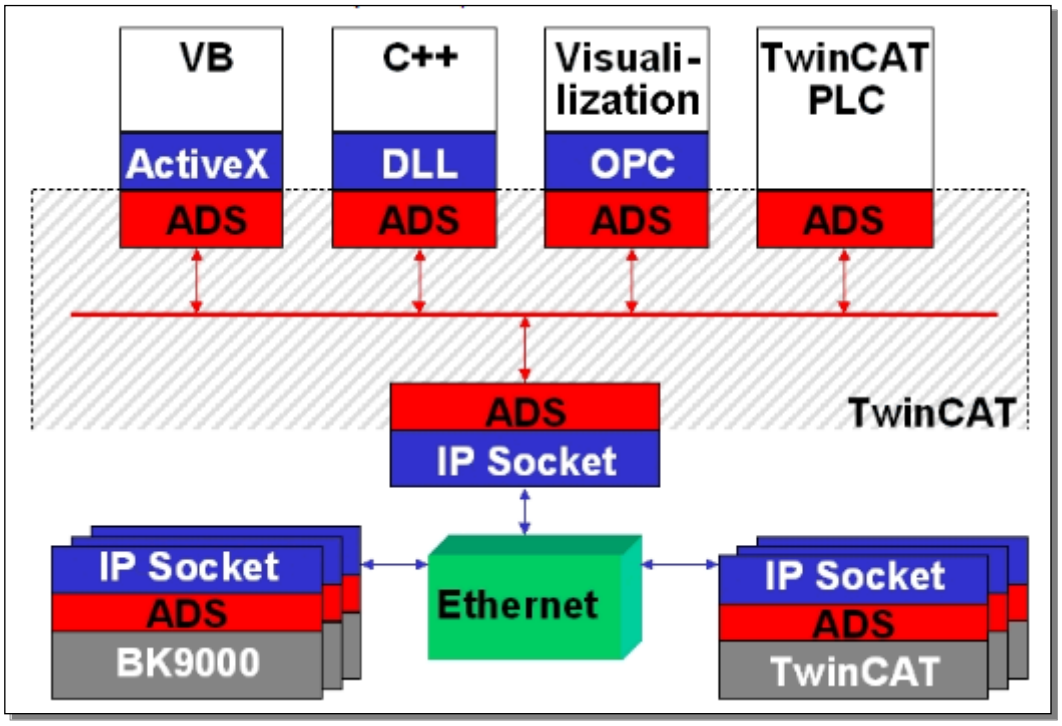

Kuva 7. Twincatin modulaarinen laitekonsepti.[3]

PLC-logiikka ohjelmamoduulissa on reaaliaika-ohjauksen ohjelmointiympäristö ja se tukee laitevalmistajista riippumattoman OpenPlc:n IEC1131-6-standardin mukaisesti kuutta ohjelmointikieltä, LD, IL; FBS, SFC, CFC ja ST. Tässä työssä käytetyt kielet ovat ST, LD ja SFC.

Tämän lisäksi pystytään MS Visual Studioon integroidussa kehitysympäristössä suorittamaan mm. Visual Basic-, C/C++/C#-ohjelmakoodia ja Matlab-ohjelmakoodia.

Twincat-järjestelmän useat palvelimet suorittavat reaaliaikaisesti eri ohjelmia ja järjestelmän analyysiä sekä konfiguraatiota. Kaikki Windows-ohjelmat, esim. visualisointisovellukset ja Office-tuotteet pääsevät käsiksi Twincatin tietoon ja voivat ohjata palvelimia Microsoft-rajapintojen kautta.

## 3.1 ADS-protokolla




Kuva 8. ADS-protokollan rajapintoja eri sovelluksiin.

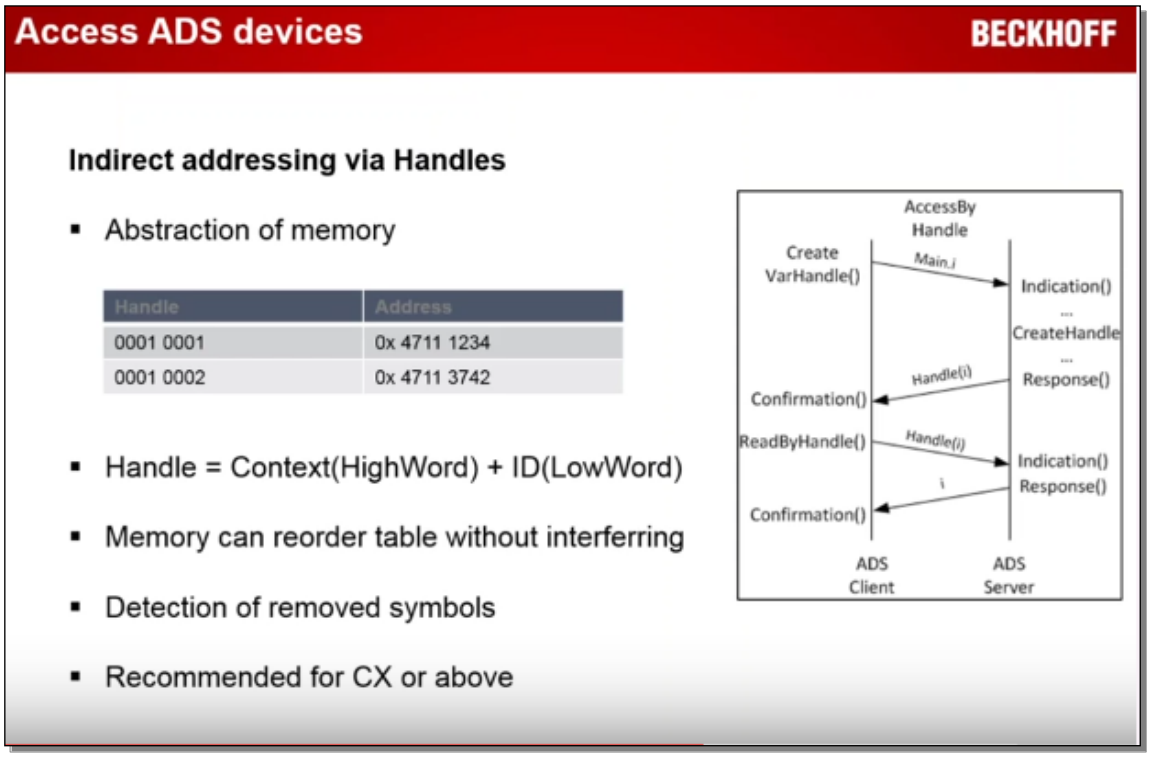
ADS-protokolla (”Automation Device Specification”) on TCP/IP-, sekä UDP/IP-standardiin perustuva kuljetuskerroksen viestipohjainen tiedonsiirtomenetelmä, jota käytetään Twincat-järjestelmän sisäiseen ja ulkoiseen kommunikointiin. Kaikki viestintä Twincat-järjestelmässä tapahtuu aina ADS-rajapinnan kautta viestin reitittimien (”message router”) toimesta. Viestin reititin on ”paikallinen postikonttori”, joka välittää ADS-viestit asianmukaisiin osoitteisiin. ADS-välittää viestin IP-verkossa fyysisestä etäisyydestä välittämättä, joko paikallisten ohjelma moduulien tai etäällä, fyysisesti eri verkossa sijaitsevien moduulien välillä. Se tarkoittaa virtuaalisina laitteina toimivien ohjelma-moduulien välistä viestintää, datan ja palvelujen vaihtoa palvelin- (”server”) ja asiakaskone (”client”) -periaatteella.[1] Viestin reititys sisäisestä järjestelmästä ulospäin etenee kuvan 8. mukaisesti Ethernet-portista TCP/IP-väylään, josta se voidaan edelleen reitittää esim. muihin Twincat-järjestelmiin tai laiteympäristöihin. Protokollana voi olla mm.

Ethercat, PROFIBUS, sarjaportti, tms. Twincatin viestin reitittimet ovat jokaisessa Beckhoffin Twincat PC:ssä ja BC-sarjan väyläpäätelaitteessa.

### 3.1.1 ADS-protokollan viestintäyhteys ja määritykset sekä palvelut

ADS tukee sekä synkronista, asynkronista että tapahtumaan perustuvaa viestintätapaa palvelin- ja asiakassovelluksen välillä. Tämän lisäksi tuettuja viestintämenetelmiä ovat osoitteen perusteella tapahtuva viestintä sekä muuttujakahvaan perustuva viestintä.

Kuvassa 9. on tietoa muuttujakahvaan perustuvan perustuvasta viestinnästä.




Kuva 9. ADS-laitteiden viestintä muuttujakahvaan perustuen.[3]

Synkroninen viestintä on syklistä, jossa viestin lähettävä isäntäkoneen (”Master”) viesti kiertää väylän alemman tason kenttälaitteet järjestäen yksi kerrallaan. Jaksotus synk-

ronisessa viestinnässä toimii siten, että viestin lähettäjänä toimiva asiakas sovellus lähettää kyselyn palvelimelle ja odottaa kunnes vastaus on valmis. Tämä on tahdistettu ohjelman suorittamiseen. Soveltamisesimerkkinä voi olla PLC:n ja käyttöliittymän (HMI) välinen viestintä.

Asynkronisessa viestinnässä asiakas lähettää kyselyn palvelimelle ja jatkaa omaa suoritustaan kunnes palvelin lähettää vastauksen takaisin. Sovellusesimerkki asynkronisesta viestinnästä voisi olla kahden PLC-moduulin välinen viestintä.

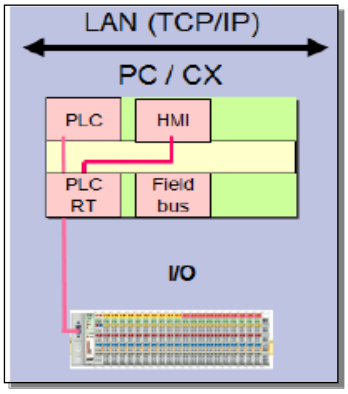
Tapahtumaan perustuva viestintä toimii ilmoitus periaatteella (”notification”). Ilmoitus voi olla syklistä tai olotilan muutokseen perustuvaa (”onChange”). Synkroniseen tai asynkroniseen viestintään verrattuna tapahtumapohjaisuus vähentää viestiliikenteen palvelupyyntöjä. Etäkokoonpanossa paikallisia muuttujia voidaan mm. puskuroida muistiin ja lähettää etäkoneelle tarkoituksenmukaisena ajankohtana.

Tapahtumapohjaisuus tarjoaa tarkan aikaleiman tiedonsiirrossa, joten se soveltuu hyvin mittaustehtäviin tai esim. HMI/SCADA-sovelluksissa käyttöliittymän visualisointi ja esim. prosessimuuttujien kuvaajan piirtämiseen. [3]

### 3.1.2 Reitin määritys ADS-laitteiden välillä

Paikallinen yhteys

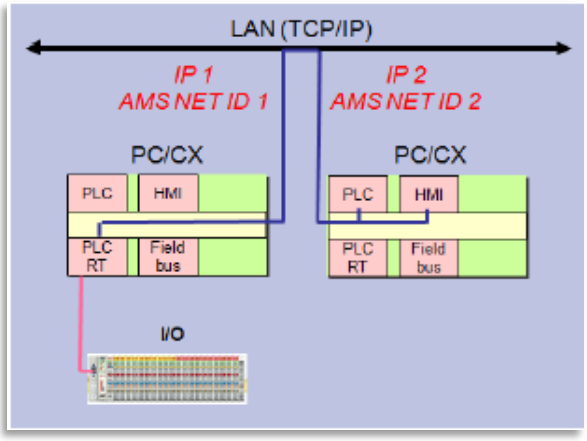
Paikallinen ympäristö näkyy kuvasta 10. Tässä PLC-yksikössä ohjaus ja kommunikointi tapahtuvat sisäisten moduuleiden, PLC RT:n, HMI-moduulin sekä I/O:n välillä. Esimerkkinä voisi olla paikallinen moottorikäyttö, jossa on paikallinen graafinen käyttöliittymä (HMI/GUI), joka operoi I/O:n liitettyä toimilaitetta, kuten servomoottoria.




Kuva 10. Paikallinen yhteys, PLC RT:n ja HMI:n välillä.[6]

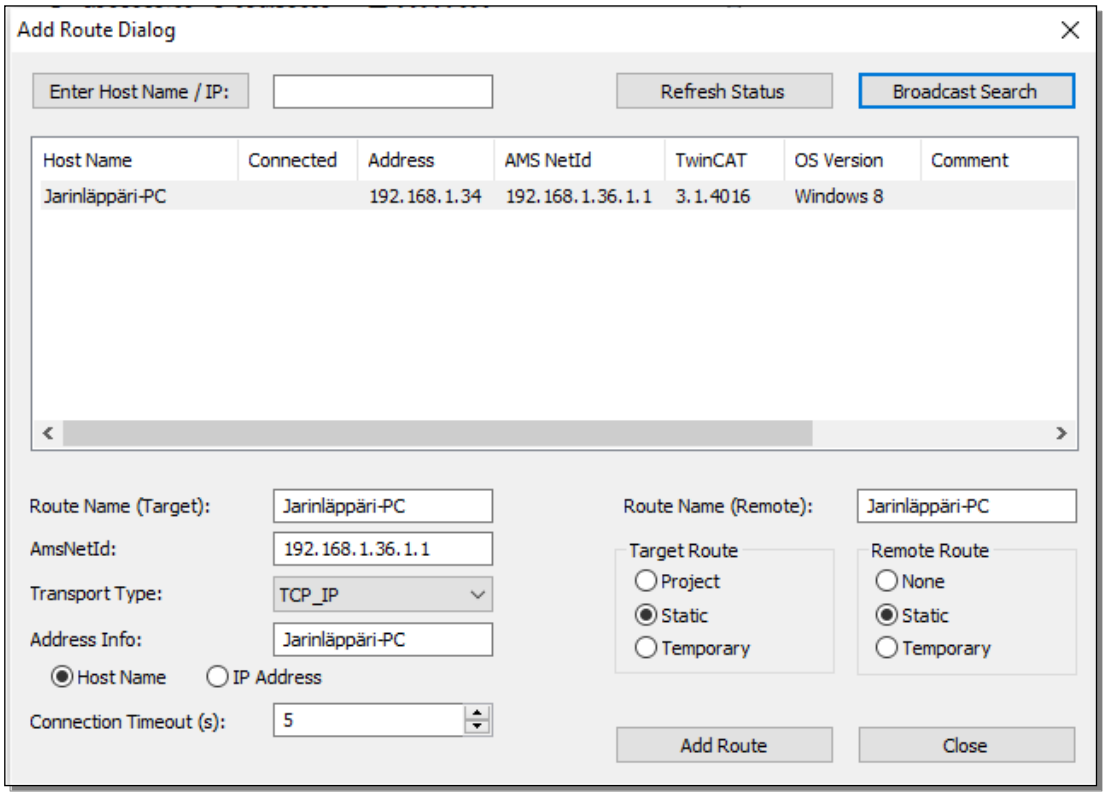
Etäyhteys kahden ADS-laitteen välillä

Kahden laitteen välinen yhteys täytyy erikseen määritellä järjestelmähallinnassa. Kuvassa 11. näkyy kahden PLC:n järjestely, jossa ne ovat yhteydessä TCP/IP-verkon välityksellä. Näin reaaliaikainen ohjaus ja tiedonsiirto ovat mahdollisia kahden PLC:n välillä. Rajoittavina tekijöinä ovat mm. siirrettävän tiedon määrä ja tietoverkon kapasiteettiin sekä nopeuteen liittyvät seikat.




Kuva 11. Etäyhteys kahden ADS-laitteen välillä IP-verkossa.[6]

Kuvassa 12. näkyy järjestelmänhallinnan määrittelydialogi reitinmäärityksestä. Reitin määrityksessä asetetaan lähde PLC ja kohde PLC. Parametreina ovat mm. ip-osoite, ”AMS NetId”-osoite.

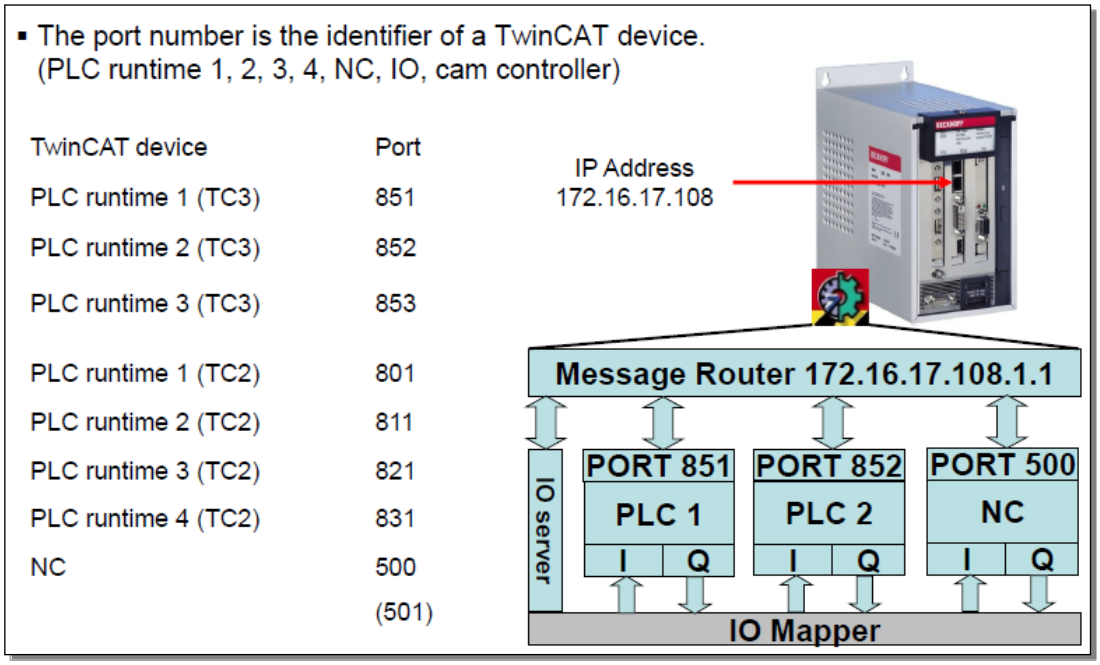



Kuva 12. Järjestelmähallinnan hallintapaneeli reitinmääritystä varten.

### 3.1.3 ADS-protokollan osoitteen muodostus sekä tiedon jäsennys ja hallinta

ADS-protokollassa kunkin viestin reitittimen perus tunnisteita ovat ”AMSNetId”-tunnus, sekä tämän alitunniste, ”AdsPort”-numero. ”AMSNetId”-tunnus on laajennos tcp/ip-osoitteesta ja vastaa muodoltaan ip-osoitetta. Tunnukseen lisätään 1.1 perään, jolloin se on ikään kuin aliverkon maski. ”AMSNetId”-tunnus on yksittäisen Twincat-

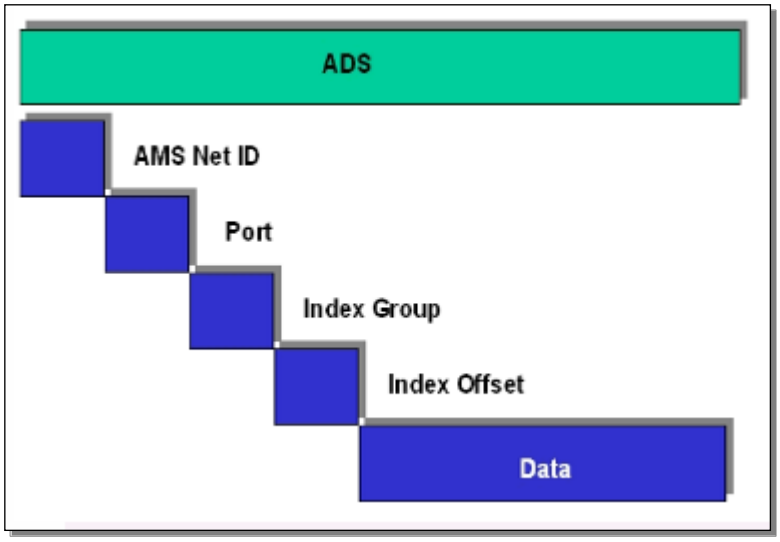
järjestelmän asennuskokoonpanon (PC,CX,BX,BC) sekä tämän viestin reitittimen tunnus. ”AdsPort”-numero kertoo Twincat-järjestelmään kuuluvan yksittäisen laiteen tunnuksen ja roolin järjestelmässä. Palvelimena toimivat Twincat PC:t, joissa voi olla lähes rajaton määrä (”Twincat 3”) reaaliaikaisia PLC-palvelimia tai esim. BC-sarjan väyläpäätelaitteita. Kuvan 12 mukaisesti ne saavat kiinteän ”AdsPort”-numeron alkaen porttinumerosta 800. Muita kiinteän porttinumeron sovelluksia ovat mm. kenttäväylien I/O-palvelimet numerosta 300 alkaen sekä numeerisen liikkeenohjaimen (NC) palvelimet numerosta 500 alkaen. Yksittäiset asiakassovellukset, kuten visualisointi sovellukset saavat muuttuvat porttinumeron saapuessaan ADS-rajapintaan.




Kuva 13. Twincatin toimilaitteiden porttien numerointi. [6]

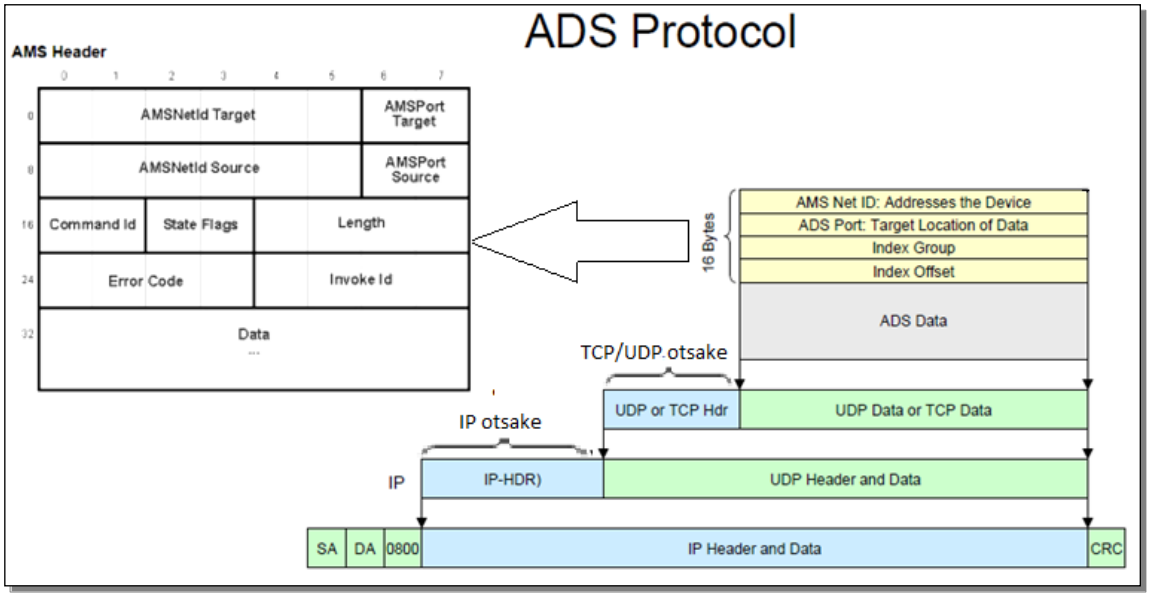
ADS:n porttinumeron aliluokkana ovat indeksiryhmä (”Index Group”) ja indeksipoikkeama (”Index Offset”)-luokat. Indeksiryhmän mitta on 16-bittiä ja indeksipoikkeaman 32-bittiä. Indeksiryhmän arvo erottelee porttinumeron datan eri ryhmiksi. Indeksipoikkeaman arvo ilmoittaa sen tavun numeron, josta lukemis- tai kirjoitus operaatio alkaa. Nämä kaksi tunnistetyyppiä ovat koko ADS-protokollan oleellisimpia parametrejä, sillä tiedon lukemien- ja kirjoittaminen tapahtuu PLC-rajapinnassa näiden muuttujien avulla.

Kuvissa 14. on yleiskatsaus ADS:n protokollapinosta.




Kuva 14. ADS-protokollapinon kuvaus.[11]

Kuvassa 15. on ADS:n rakenteesta tarkempaa tietoä ja sen lomittumisesta TCP/IP-protokollaan. Ylinpänä ”AMS Header” -otsakkeen osiossa ”AMS Net ID” -osoite lähettäjälle ja vastaanottajalle.




Kuva 15. ”AMS Header”-otsakkeen lähettäjä- ja vastaanottajatietoja ADS-protokollassa.

### 3.1.4 ADS-kirjoitus- ja lukuoperaatiot PLC-tasolla ja IEC61131-3-kielellä

ADS-tiedon operaatiot prosessikuvassa jaetaan PLC-muistialueen muuttujiin (%M-alue) ja TwinCAT ADS-järjestelmäpalveluiden muistialueen tulopuolen (%I-alue) ja lähtöpuolen (%Q-alue) muuttujiin.

Indeksiryhmäparametri (”index group”) kuvaa ja luokittelee ADS-laitteen roolia järjestelmässä. Indeksipoikkeama (”index offset”) ilmaisee varsinaisen osoitteen kullakin eri muistialueella. Kuvien 16 ja 17 taulukoissa esitetään ADS-muistipaikan osoitteen muodostumista ja minkälaisia ADS-operaatiota ko. muistipaikkaan talletulle muuttujalle saa suorittaa. Taulukon esimerkeissä käydään läpi vain luku- ja kirjoitusoperaatioita. Kuvan 19. taukossa luetellaan ADS palvelun kaikki yhdeksän operaatiota.

| PLC:n ADS palvelut | | | |
|---|---|---|---|
| PLC Lähde/ Kohde muistialue muuttujille | INDEKSI RYHMÄ | INDEKSI POIKKEAMA | ESIMERKKI |
| READ_M / WRITE_M. Lukee/Kirjoittaa PLC muistiin kenttään %M. Poikkeama on tavuina. | 0x4020 | 100 | Määritellään kokonaislukutyyppinen muuttuja ”iVar1” muistiosoitteeseen ”%MB100”.<br>VAR<br>iVar1 AT %MB100: INT;<br>END_VAR<br>IDXGRP 0x4020<br>IDXOFFS 100 |
| READ_MX / WRITE_MX. Lukee/Kirjoittaa PLC muistiin kenttään %MX. Poikkeama on bitti-osoite. Tämä lasketaan: tavu-osoite*8+bitti-osoite. | 0x4021 | 81 | Määritellään binäärityyppinen muuttuja ”bVar1” muistiosoitteeseen ”%MB10.1”.<br>VAR<br>bVar1 AT %MX10.1: BOOL;<br>END_VAR<br>IDXGRP 0x4021<br>IDXOFFS 81 |

Kuva 16. PLC:n ”%M-muistialueen osoitteita ADS PLC-palveluille.[9]

| Yleiset Twincat ADS järjestelmäpalvelut | | | |
|---|---|---|---|
| PLC Lähde/ Kohde prosessi | INDEKSI RYHMÄ | INDEKSIP OIKKEAMA | ESIMERKKI |

| alueen tulo- ja lähtö muuttujille | | | |
|---|---|---|---|
| READ_I / WRITE_I. Lukee/Kirjoittaa PLC prosessin fyysisiä tulomuuttujia %I kentässä. Poikkeama on tavuina | 0xF020 | 0 | Määritellään kokonaislukutyyppinen tulomuuttuja ”Ain1” kentässä ”%IB0”.<br>`VAR`<br>`  Ain1 AT %IB0: INT;`<br>`END_VAR`<br>`IDXGRP 0xF020`<br>`IDXOFFS 0` |
| READ_IX / WRITE_IX. Lukee/Kirjoittaa PLC prosessin fyysisiä tulomuuttujia %IX kentässä. Poikkeama on bitti-osoite. Tämä lasketaan: tavu-osoite*8+bitti-osoite. | 0xF021 | 41 | Määritellään binäärityyppinen tulomuuttuja ”Iin1” kentässä ”%IB0”.<br>`VAR`<br>`  Ini1 AT %IX5.1: BOOL;`<br>`END_VAR`<br>`IDXGRP 0xF021`<br>`IDXOFFS 41` |
| READ_Q / WRITE_Q. Lukee/Kirjoittaa PLC prosessin fyysisiä lähtömuuttujia %Q kentässä. Poikkeama on tavuina | 0xF030 | 100 | Määritellään kokonaislukutyyppinen lähtömuuttuja ”iVar1” muistiosoitteeseen ”%MB100”.<br>`VAR`<br>`  VeloStepper AT %QB100: INT;`<br>`END_VAR`<br>`IDXGRP 0xF030`<br>`ISXOFF 100` |

| | | | |
|---|---|---|---|
| READ_QX / WRITE_QX. Lukee/Kirjoittaa PLC prosessin fyysisiä lähtömuuttujia %QX kentässä. Poikkeama on bitti-osoite. Tämä lasketaan: tavu-osoite*8+bitti-osoite. | 0XF011 | 167 | Määritellään binäärityyppinen lähtömuuttuja ”Dout1” muistipaikkaan ”%QX20.7”.<br>VAR<br>Dout1 AT %QX20.7: BOOL;<br>END_VAR<br>IDXGRP 0xF011<br>IDXOFFS 167 |

Kuva 17. PLC:n tulopuolen ”%I” ja lähtöpuolen ”%Q” muistialueiden osoitteita ADS-järjestelmäpalveluille.[9]

### 3.1.5 ADS-protokollan palvelujen metodit ja toteutukset eri ohjelmointikielillä

ADS-metodien toteutus tapahtuu aina PLC-rajapinnassa. Twincat-kehitysympäristö tukee IEC61131-3:n PLC-järjestelmäkirjaston lisäksi korkeampien ohjelmointikielien dll-komponenttikirjastoja. Tuettuja ohjelmointikieliä ja niiden ADS-komponenttikirjastoja esitellään kuvan 18 taulukossa.

| Language | ADS Component |
|---|---|
| PLC (IEC61131-3) | TcSystem Library |
| .NET (C#, VB) | TwinCAT.Ads.dll |
| C/C++ | TcAdsDll |
| Delphi | AdsOcx |
| JavaScript | TcAdsWebService |
| Java | AdsToJava |
| Others | Support via Wrapper or Webservice |

Kuva 18. Luettelo ADS-kirjastoja tukevista ohjelmointikielistä. [3]

ADS-protokollala on kuva 19. mukaisesti yhdeksän metodia, joilla toteutetaan palveluja. Samat palvelut on määritelty kaikille edellisen kuvan 18. taulukon kielille.

| Id | Service | Purpose | Use case |
|---|---|---|---|
| 1 | ReadDeviceInfo | Provide device info | Get TwinCAT Version and Buildnr |
| 2 | Read | Read access | Read variable |
| 3 | Write | Write access | Write variable |
| 4 | ReadState | Get device state | Get TwinCAT State (Run/Config) |
| 5 | WriteControl | Change device state | Switch TwinCAT (Config ←→Run) |
| 6 | AddDeviceNotification | Annouce a notification | Get variables on change |
| 7 | DeleteDeviceNotification | Delete a notification | Delete the announcement |
| 8 | DeviceNotification | Notification | Get timestamp and changed data |
| 9 | ReadWrite | Read Write access | Provide handle for read access |

Kuva 19. Luettelo ADS-palveluiden käskyistä/metodeista.[3]

### 3.1.6 Yhteyden muodostaminen ADS-laitteiden välille PLC-tasolla

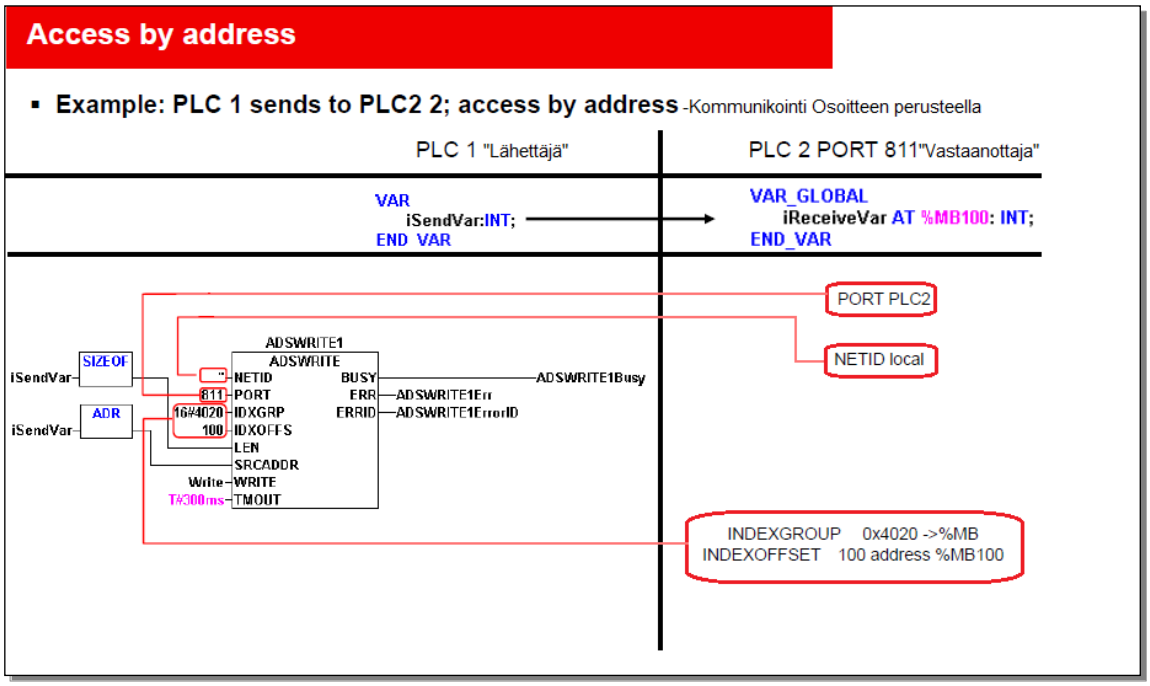
Yhteyden muodostus kahden PLC:n välillä voidaan toteuttaa seuraavasti:

1. Osoitteen perusteella tapahtuva kommunikaatio.
2. Muuttujakahvan perusteella tapahtuva kommunikaatio.

Osoitteen perusteella tapahtuva viestintä

Kuvassa 20. näkyy osoitteeseen perustuva kommunikointi “ADSWRITE”-funktioblokilla. Kuvassa lähettävän osapuolen ”PLC1”:n ”%M”-muistialueen kokonaislukumuuttujaan ”iSendVar” on tallennettu kirjoittavan operaation ADS-funktioblokki ”ADSWRITE1”. Funktioblokin tulopuolen parametreiksi on määritelty indeksiryhmän arvo ”0x4020” ja indeksipoikkeaman arvo ”%MB100”, joka on varsinainen viestin vastaanottajan osoite.

Viestin vastaanottavan osapuolen, ”PLC 2 PORT 811”:n PLC-ohjelmaan on määritelty globaaleihin muuttujiin kokonaislukutyyppinen muuttuja ”iReceiveVar osoitteeseen ”%MB100”. Näin vastaanottava ADS-laite pystyy lukemaan muuttujan ”iReceiveVar” kirjoitettua tietoa.

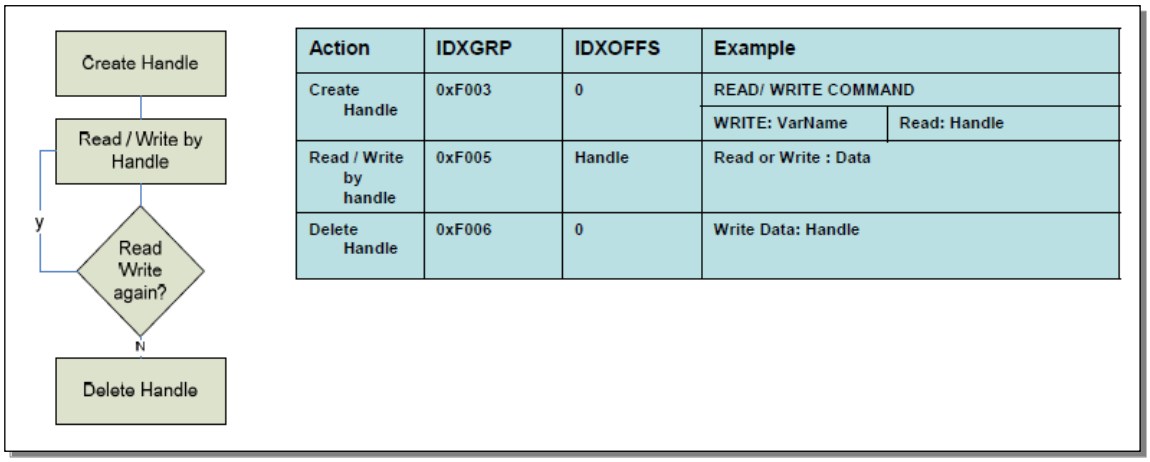



Kuva 20. Osoitteen perusteella tapahtuva viestintä kahden ADS-laitteen välillä.[9]

Muuttujakahvan perusteella tapahtuva viestintä

Muuttujakahvaan perustuva viestintä etenee kuvan 21. mukaisen vuokaavion mukaisesti kolmessa vaiheessa:

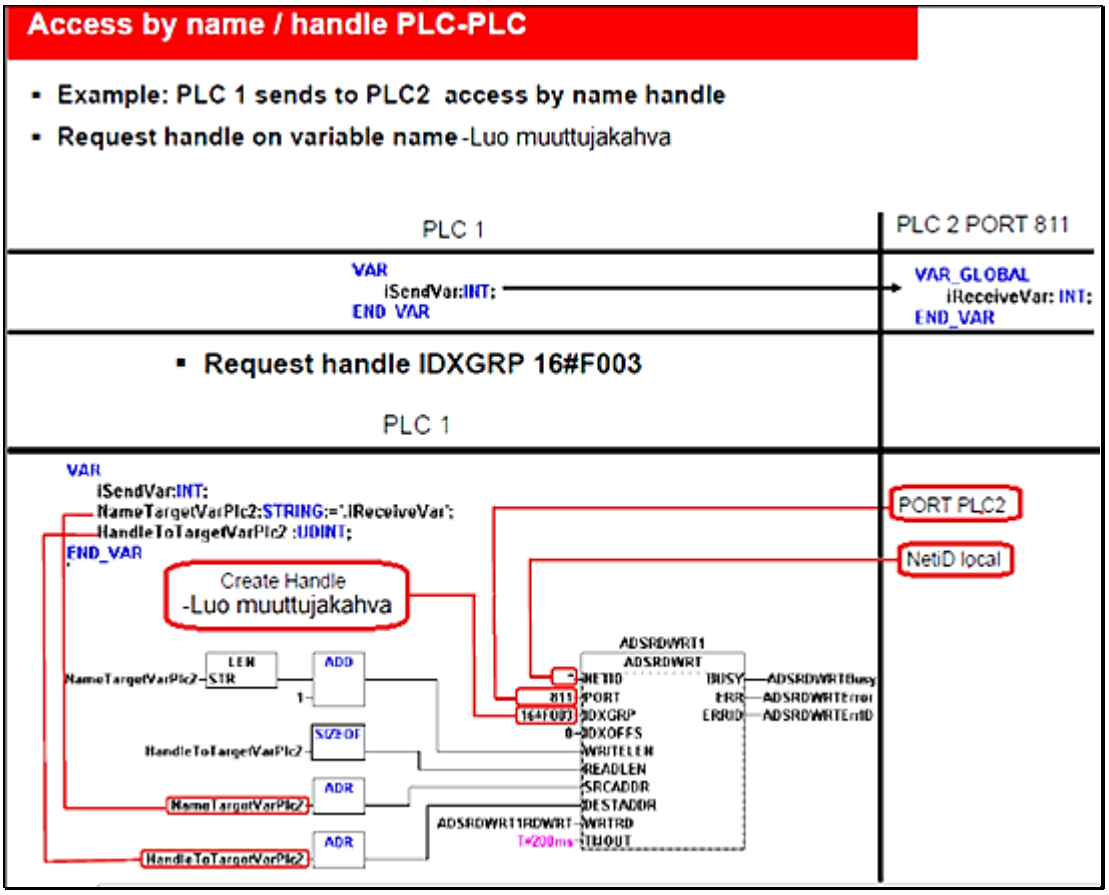
1. Luodaan muuttujakahva. Indeksiryhmän arvo on (0xF003) ja indeksipoikkeaman arvo(0).
2. Kirjoitetaan tai luetaan dataa. Indeksiryhmän arvo on (0xF005) ja indeksipoikkeaman arvo (Kahvamuuttuja).
3. Poistetaan kahvamuuttuja. Indeksiryhmän arvo on (0xF006) ja indeksipoikkeaman arvo(0).




| Action | IDXGRP | IDXOFFS | Example | |
|---|---|---|---|---|
| Create Handle | 0xF003 | 0 | READ/ WRITE COMMAND | |
| | | | WRITE: VarName | Read: Handle |
| Read / Write by handle | 0xF005 | Handle | Read or Write : Data | |
| Delete Handle | 0xF006 | 0 | Write Data: Handle | |

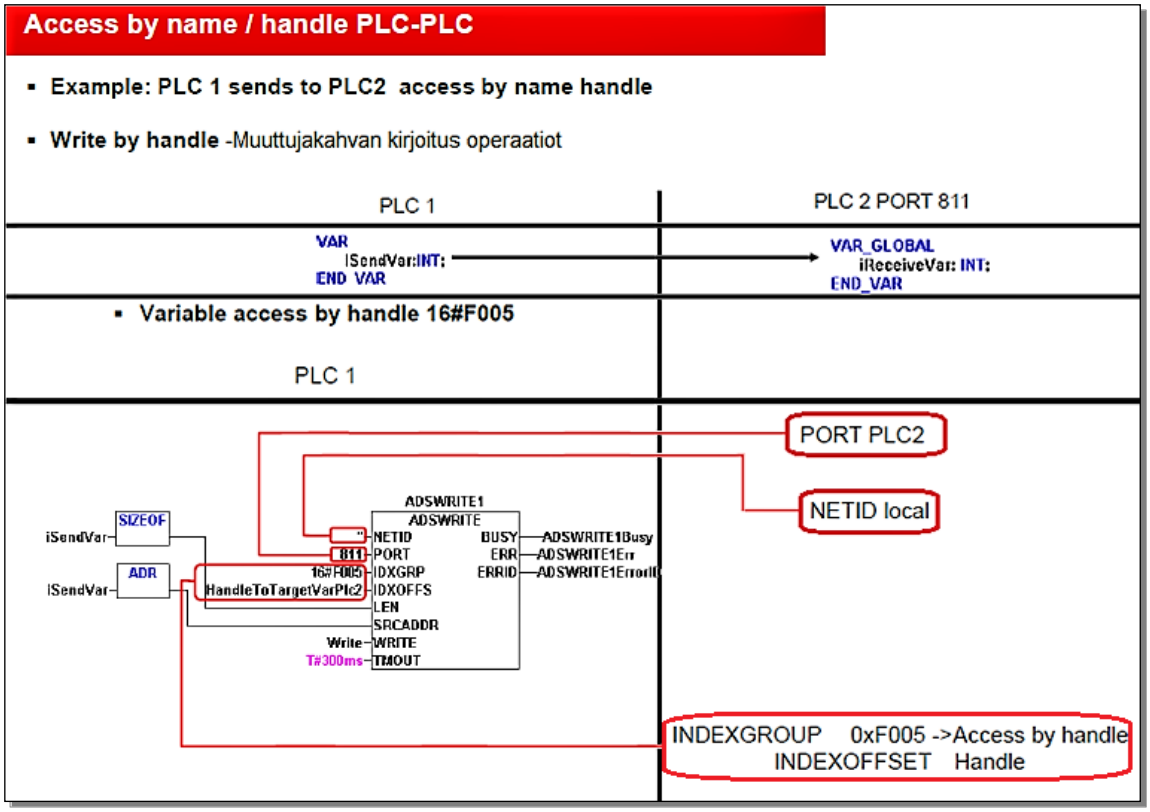
Kuva 21. Vuokaavio muuttujakahvaan perustuvan viestinnän eri vaiheista. [9]

Kuvassa 22. nähdään viestinnän ensimmäisessä vaiheessa muuttujakahvan luonti. Kuvassa ”PLC 1” toimii lähettäjänä ja ”PLC 2 PORT 811” vastaanottajana. Lähettäjän tyyppi on kokonaislukumuuttuja ”iSendVar”, johon ADS-funktionblokin kirjoitusmetodi ”ADSDWRT1” on talletettu. Metodi kirjoittaa vastaanottajan kokonaislukumuuttujaan ”iReceiveVar”, kahvamuuttujalla ”HandleToTargetVarPlc2”



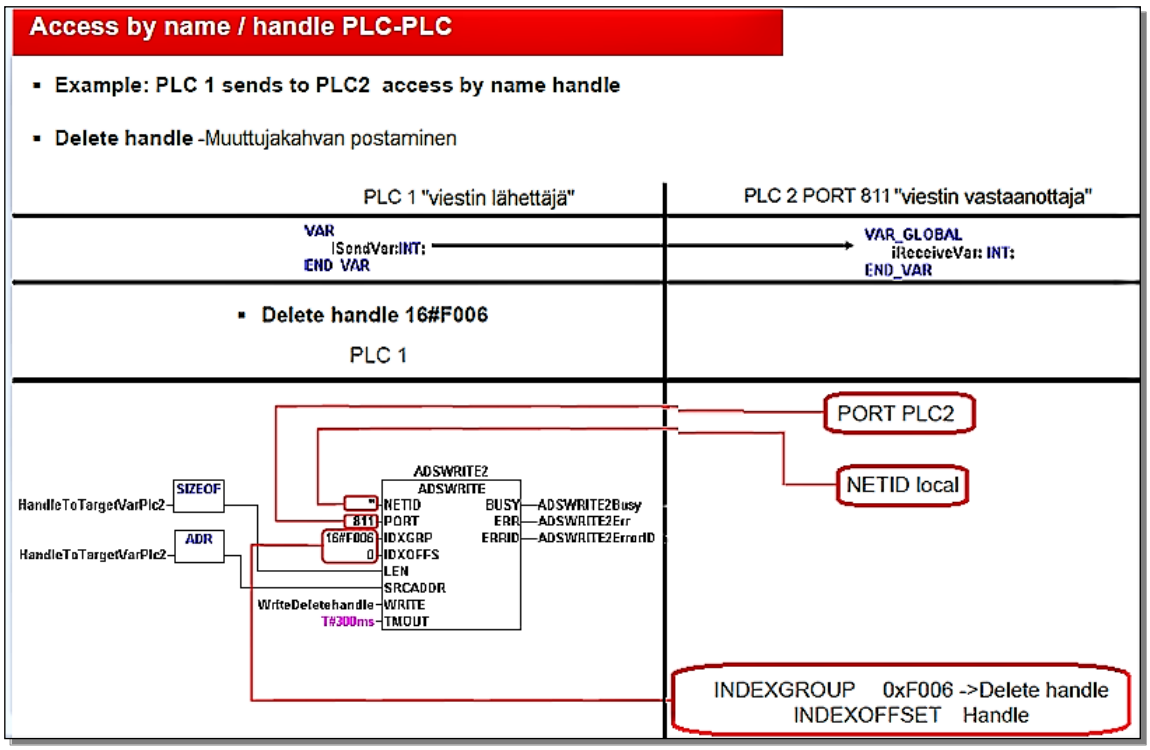

Kuva 22. Muuttujakahvan ”HandleToTargetVarPlc2” luonti ”ADSDWRT1”-operaation parametrilla ”16#F003”. [9]

Kuvassa 23. nähdään viestinnän toisessa vaiheessa kirjoitusoperaatio muuttujakahvaan funktioblokilla ”ADSWRITE1”, jossa tuloparametreina on indeksiryhmän arvona ”16#F005” ja indeksipoikkeaman arvona äskettäin luotu kahvamuuttuja ”HandleToTargetVarPlc2”




Kuva 23. Muuttujakahvan kirjoitus operaatio ”ADSWRITE1” parametrilla ”16#F005”. [9]

Kuvassa 24. nähdään viestinnän kolmannessa ja viimeisessä vaiheessa lähettävän PLC1:n kirjoitusoperaatio muuttujakahvaan funktioblokilla ”ADSWRITE2”, jossa tuloparametreina oleva indeksiryhmän arvo ”16#F006” poistaa luodun kahvamuuttujan joka on indeksipoikkeaman arvona edelleen ”HandleToTargetVarPlc2”.




Kuva 24. Muuttujakahvan poistooperaatio ”ADSWRITE2” parametrilla ”16#F006”. [9]

### 3.1.7 Yhteyden muodostaminen ADS-laitteiden välille .NET-ohjelmointiympäristössä

ADS:n kaikki tarvittavat tietorakenteet ja tyypit, kuten luokat ja struktuurit kuuluvat .NET C#-ohjelmoinnissa nimiavaruuteen TwinCAT.Ads. Luokka ”TcAdsClient” toimii ”käärijä-luokkana” tai säiliöluokkana TwinCAT.Ads-luokalle ja mahdollistaa kommunikoinnin muiden ADS-laitteiden kanssa. Kommunikoinnin metodi on ***”Connect()”,*** jonka ottaa parametreinaan netID-tunnisteen ja/tai srvPort-tunnisteen. NetId on ADS-palvelimen ”AMSNetId”-tunnus *ja* ”srvPort” on ADS-palvelimen porttinumero.

AdsStream-luokka

“AdsStream”-luokka on tietovuoluokka, jota käytetään ADS-viestintään. Se periytyy ”System.IO.Memorystream”-luokasta, joka tallentaa keskusmuistiin.

”AdsStream”-luokkaan voi kirjoittaa käyttämällä ”System.IO.BinaryWriter”-luokan metodeja ja sitä voidaan lukea käyttämällä ”System.IO.BinaryReader”-luokan metodeja. Kuvassa 25. on esimerkki C#-sovelluksesta, joka käyttää ”AdsStream”-luokan objektia PLC-muuttujan lukemiseen. Lukumetodina on ”BinaryReader”-luokan metodi, joka käyttää luotua ”AdsStream”-luokan objektia parametrina. Sovellus lukee kokonaislukutyyppisen, 32-bittisen PLC-muuttujan käyttöliittymän ”textBox1”-objektin tekstikenttään.[4]

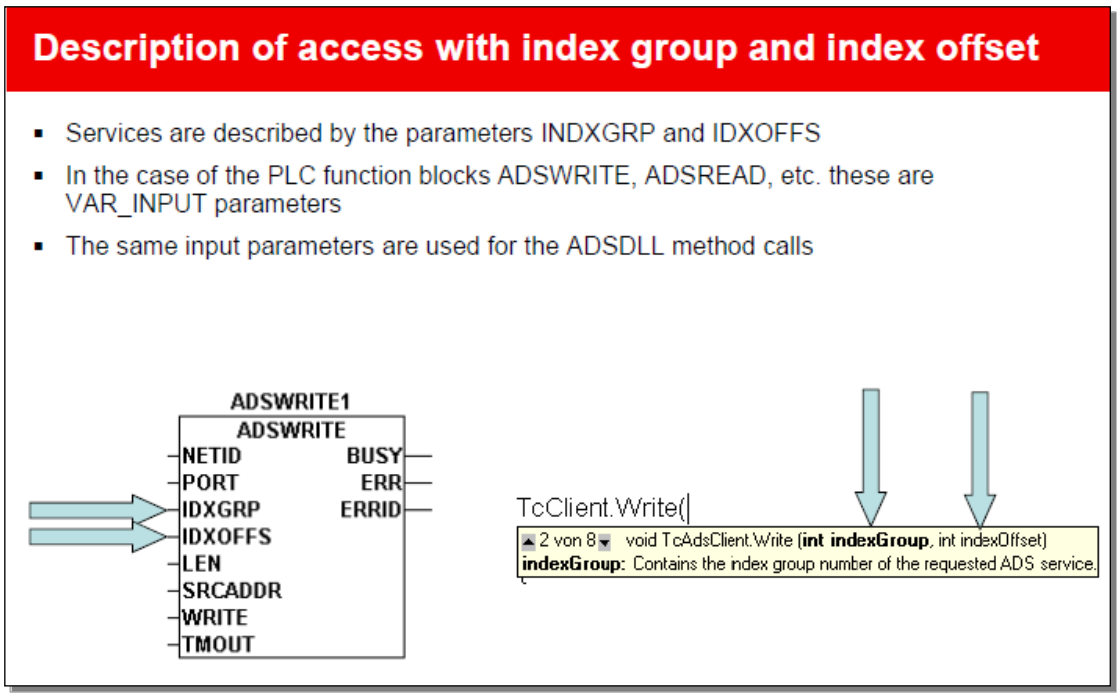
**Reading a PLC variable:**

```
// creates a stream with a length of 4 byte
AdsStream ds = new AdsStream(4);
BinaryReader br = new BinaryReader(ds);

// reads a DINT from PLC
tcAds.Read(0x4020,0,ds);

ds.Position = 0;
textBox1.Text = br.ReadInt32().ToString();
```

Kuva 25. PLC-muuttujan luku-sovellus C#-kielellä [4]

ADS-viestintä PLC-sovelluksen (IEC61131-3) JA .NET C#-sovelluksen välillä

Ohjelmamoduulien välinen ADS-metodien käyttö ja vastaavuus korkeamman luokan ohjelmointikielten, kuten .NET:n C#-kielen ja laiteläheisten PLC-ohjelmointikielten välillä on Twincat-ohjelmointiympäristössä järjestetty indeksiryhmä- ja poikkeamaparametrien ympärille. Kuvassa 26 vasemmalla puolella on IEC61131-6:llä ilmaistuna funktioblokki ”ADSWRITE”-metodi, eli muuttujan ADS-kirjoitus metodi. Tässä indeksiryhmä- ja poikkeamaparametrit asetetaan funktioblokin tulopuolelle. Oikeassa reunassa on vastaava operaatio ilmaistuna C#-kielen metodilla ”***void TcClient.Write(int IndexGroup, int IndexOffset)***”.




Kuva 26. ADS-kirjoitusmetodi IEC61131-3-funktioblokkina ja :NET C#-kielellä.[9]

.NET C#-kehitysympäristön ADS-metodikutsut

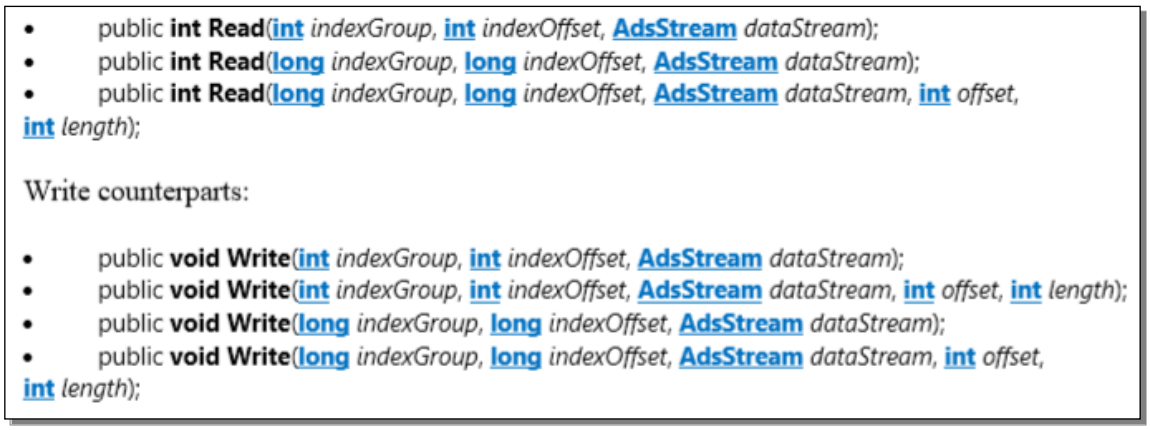
Seuraavassa käydään läpi ”.NET”-kehitysympäristön C#-kielen luku- ja kirjoitusmetode-ja. Metodit toteutetaan joko tietyn luokan objektiin sidottuina tai ”AdsStream”-tietovuoluokkaan sidottuina.

ADS-metodien kutsu osoitteen, eli indeksiryhmän ja indeksipoikkeaman perusteella.

Kuvan 27. luku- ja kirjoitusmetodit lukevat tai kirjoittavat data synkronisesti ADS-laitteista ja tallentavat tuotoksensa objektiin.


Kuva 27. C#-kielellä tehtyjä ADS- luku- ja kirjoitusmetodeja, jotka ovat objektiin sidottuja. [6]

Kuvassa 28. on luetteloitu ”AdsStream” tietovuoluokkaan sidottuja ADS- luku- ja kirjoitusmetodeita.




Kuva 28. C#-kielellä tehtyjä ADS- luku- ja kirjoitusmetodeja, jotka ovat ” AdsStream”-luokkaan sidottuja. [6]

Metodeiden käyttö tapahtumaperusteisten tila-ilmoitusten yhteydessä

Tapahtumaperusteisessa viestinnässä C#-kielessä PLC-muuttujia operoidaan ”Event” -tapahtuman metodiosoittimilla. Tapahtumiin perustuvaa sovellusta lähdetään luomaan seuraavan listan toimenpiteiden mukaisesti:

1. Luodaan tapahtumiin reagoiva metodi jokaiselle muuttujakahvalle. Tällainen metodi on muotoa:

```
”TcClient.AddDeviceNotificationEx(…);”
```

Tämä sijoitetaan muuttujakahvaan ”hnReady”:

```
”hnReady = TcClient.AddDeviceNotificationEx(
".Ready",
AdsTransMode.OnChange,
100, 0,
lblReady,
typeof(bool)
);”
```

2. Luodaan tapahtumakäsittelijä jokaiselle muuttujakahvalle.

   Jokainen tapahtuma luo tapahtumankäsittelijän metodin seuraavalla tavalla:

```
”  TcClient.AdsNotificationEx += new AdsNotificationExEventHandler(…)”.
```

   Tässä itse tapahtuma “event” on “***TcClient.AdsNotificationEx***”.

3. Luodaan tapahtuman funktio, joka on edellisen kohdan tapahtumakäsittelijän argumentti.

```
“void TcClient_AdsNotificationEx(object sender, AdsNotificationExEven-
tArgs e)
{


if (e.NotificationHandle == hnReady )
{
     Textbox txtbx = (TextBox)e.UserData;
     }
};”
```

   **”object sender”** on lähettäjän objekti ja objekti “**e”** pitää sisällään lähetettävän datan sekä tapahtuman aikaleiman.

   Ehtolause “***if (e.NotificationHandle == hnReady){};”*** vertailee muuttujakahvan “hnReady” tilaa.

4. Poistetaan tilailmoitukset, jotka lukevat kutakin muuttujakahvaa.

```
”TcClient.DeleteDeviceNotification(hnReady );”
```

Kuvassa 29. on luettelo objektiin sidotuista, tapahtumiin reagoivista ”AddDeviceNotificationEx()”-metodeista, jotka on liitetty C#-sovelluksessa kutakin PLC-muuttujaa vastaavaan kahvamuuttujaan. Kuvien 29. ja 30. metodeilla luodaan siis yhteys C#-sovelluksen ja PLC-muuttujien välille. Metodien parametri ”AdsTransMode” määrittävät ADS-kommunkaation muodon. Kommunikaatiomuoto voi olla esim. syklistä tai muuttujan tilan muutokseen perustuvaa (”OnChange”).

- public **int AddDeviceNotificationEx**(long indexGroup, long indexOffset, AdsTransMode transMode, int cycleTime, int maxDelay, object userData, Type type);
- public **int AddDeviceNotificationEx**(long indexGroup, long indexOffset, AdsTransMode transMode, int cycleTime, int maxDelay, object userData, Type type, int[] args);
- public **int AddDeviceNotificationEx**(string *variableName*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*, Type *type*);
- public **int AddDeviceNotificationEx**(string *variableName*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*, Type *type*, int[] *args*);

Kuva 29. Luettelo objektiin sidotuista, tapahtumiin reagoivista metodeista. [6]

Kuvassa 30. on luettelo ”AdsStream” tietovuoluokkaan sidotuista, tapahtumiin reagoivista metodeista.

- public **int AddDeviceNotification**(int *indexGroup*, int *indexOffset*, AdsStream *dataStream*, int *offset*, int *length*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*);
- public **int AddDeviceNotification**(int *indexGroup*, int *indexOffset*, AdsStream *dataStream*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*);
- public **int AddDeviceNotification**(long *indexGroup*, long *indexOffset*, AdsStream *dataStream*, int *offset*, int *length*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*);
- public **int AddDeviceNotification**(long *indexGroup*, long *indexOffset*, AdsStream *dataStream*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*);
- public **int AddDeviceNotification**(string *variableName*, AdsStream *dataStream*, int *offset*, int *length*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*);
- public **int AddDeviceNotification**(string *variableName*, AdsStream *dataStream*, AdsTransMode *transMode*, int *cycleTime*, int *maxDelay*, object *userData*);

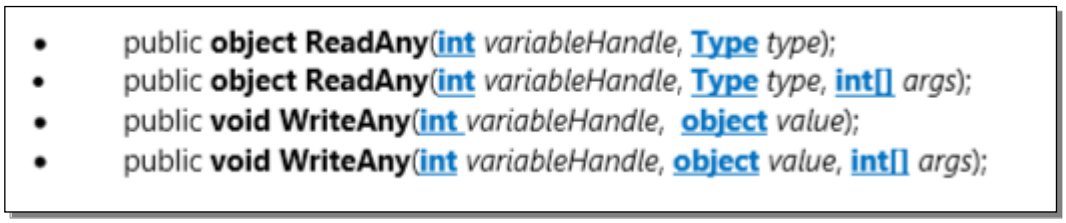
Kuva 30. Luettelo ”AdsStream”-tietovuoluokkaan sidotuista, tapahtumiin reagoivista metodeista. [6]

Metodeiden käyttö nimen, eli muuttujakahvan perusteella

Muuttujakahvojen luonti ja käyttö on seuraavan luettelon mukaista:

1. Lisää TcClient-luokka.
2. Lisää nimiavaruus "Namespace".
3. Luo instanssi objekti TcClientille.
4. Luo kahvamuuttujat.
5. Kutsu PLC-symbolimuuttujien muuttujakahvoja.
6. Kirjoita ja lue muuttujakahvalla.
7. Poista muuttujakahvat.
8. Vapauta resursseja.

C#-kieliset luku- ja kirjoitusmetodit, jotka on sidottu tiettyyn luokan objektiin näkyvät kuvan 31 luettelossa.




Kuva 31. Luettelo tiettyyn luokan objektiin sidotuista luku- ja kirjoitusmetodeista. [6]

C#-kieliset luku- ja kirjoitusmetodit, jotka on sidottu tiettyyn ”AdsStream” tietovuo-luokkaan näkyvät kuvan 32 luettelossa.


Kuva 32. Luettelo ”AdsStream” -tietovuoluokkaan sidotuista luku- ja kirjoitusmetodeista. [6]

# 4 Servomoottorikäytön toimilaitteet ja toimielimet

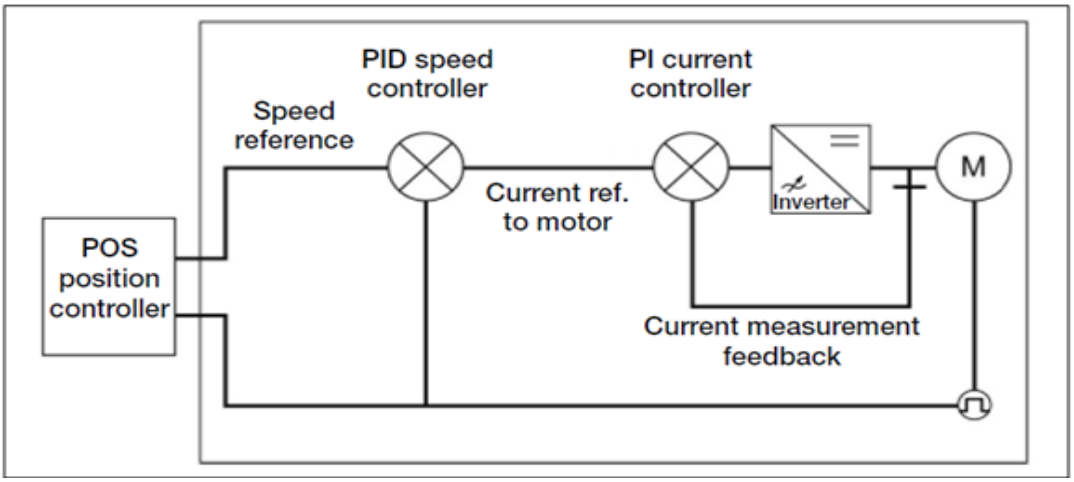
Seuraavassa käydään läpi servomoottorikäytön toimilaitteet ja komponentit, sekä niiden soveltaminen tehtyyn työhön.

Servomoottori on pyörivän liikkeen tai lineaarisen liikkeen luova toimilaite, joka mahdollistaa tarkan säädön pyörimiskulmalle tai lineaariselle paikantamiselle, nopeudelle ja kiihtyvyydelle.

Servomoottorikäyttö koostuu viidestä pääkomponentista: 1. Sähkömoottorista, 2. Pulssianturista, joka on liitetty suljetulla takaisinkytkennällä paikka ja nopeus informaation lukemiseen, 3. Automaatioväylästä, joka mm. hoitaa takaisinkytkennän sekä muun viestinnän eri toimilaitteiden välillä, 4. Servo-ohjain/liikkeenohjain yksiköstä, 5. Servovahvistin yksiköstä.

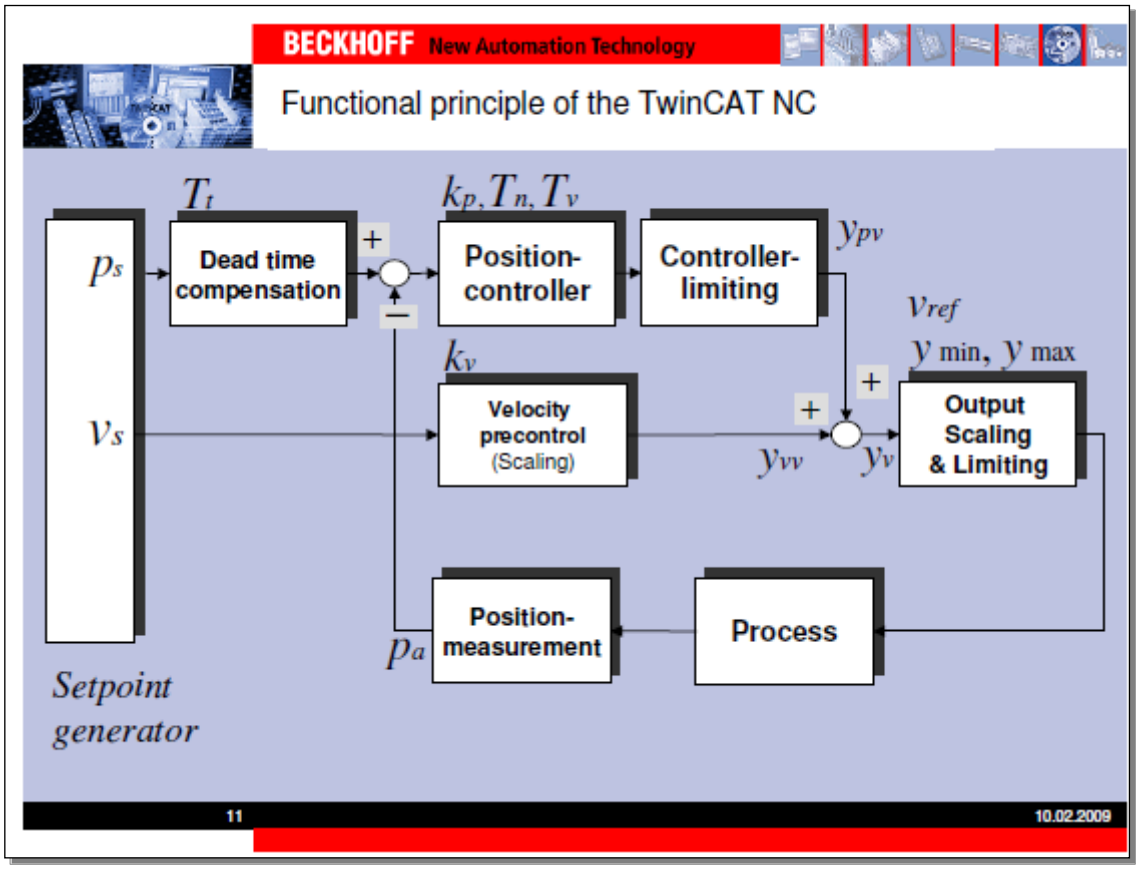
## 4.1 Servo-ohjain (NC-liikkeen ohjaimen säädön algoritmit)

Kestomagnetisoituja synkronisia moottoreita käytettäessä moottorin servovahvistinyksikön ohjauksen säätöalgoritmi säätää staattorin virtaa, joka määrittää halutun vääntömomentin ja pyörimisnopeuden. Kuvassa 33. näkyy esimerkki PID nopeussäätimen säätömallista, jossa on PI säädetty moottorin virran ohjaus ja takaisin kytketty moottorin virran mittaus.




Kuva 33. Esimerkki PID-säädetystä nopeuden ohjaimesta, jossa on PI-säädetty moottorin virran ohjaus (ABB Oy).

Kuvassa 34. on Beckhoffin käyttämä ohjelmallinen liikkeen ohjauksen (NC) malli paikan ja nopeuden ohjaukseen. Kyseessä on palvelin-ohjelma ”NC System Server”.



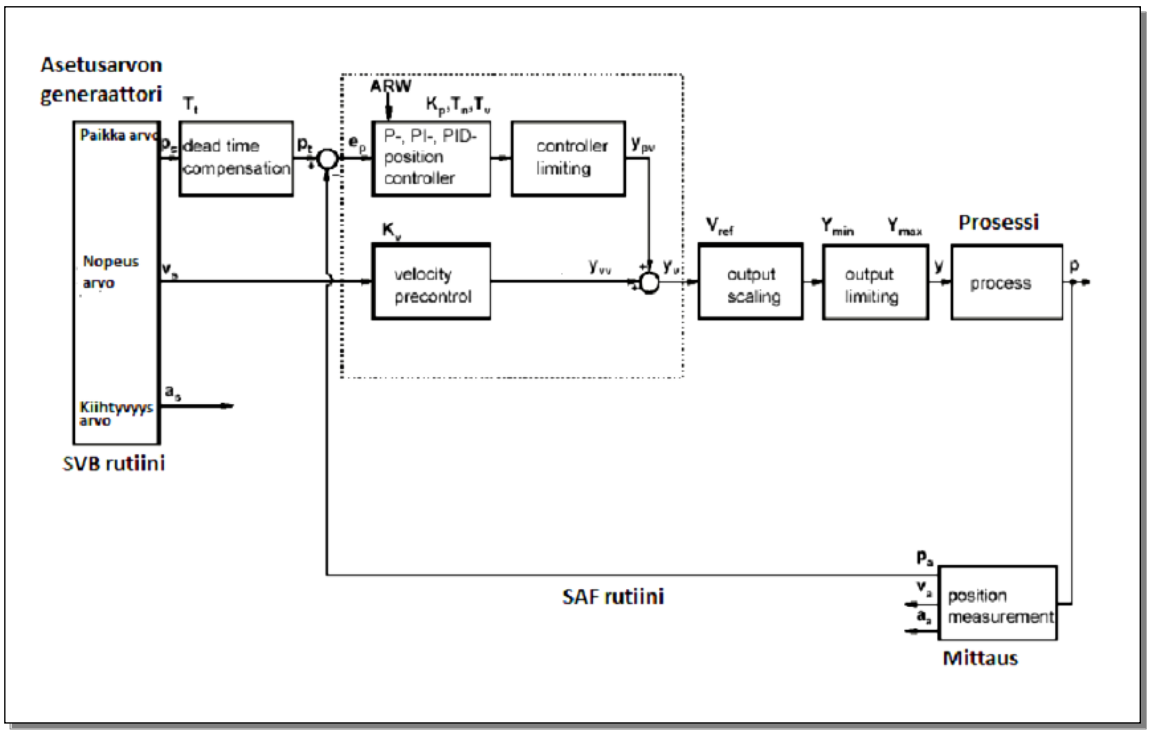

Kuva 34. Twincat-liikkeen ohjaimen (NC) toiminnallinen malli (Beckhoff Automation Oy).

”NC-System Server” -liikkeen ohjaimessa on kaksi syklistä rutiinia (”Task”), SVB ja SAF.

SVB-rutiini on liikkeen ohjauksen asetusarvon generaattori (”Setpoint generator”). Se luo paikan ohjauksen ja nopeuden ohjauksen profiilit koko järjestelmässä. Parametreina SVB:llä on mm. paikan oloarvo, paikan asetus-arvo, suurin nopeus, tasaisen kiihtyvyyden ja hidastuvuuden arvot sekä hetkittäinen kiihtyvyys. SVB-rutiinia suoritetaan tyypillisesti 10 ms:n välein ja muutokset missä tahansa yksittäisessä parametrissa aiheuttavat koko profiilin uusimiseen muuttuneiden arvojen mukaiseksi.

SAF-rutiini on prosessin mittauksen eli moottorin arvojen takaisinkytkennän säätösilmukka. Se säätää ja määrittää paikan arvot käytölle. SAF-rutiinia suoritetaan tyypillisesti 2 ms:n välein. Suoritusnopeus on mahdollista nostaa jopa 100 us:iin mikäli järjestelmän tiedonsiirto ja vahvistinyksikkö pystyvät näihin arvoihin. SAF-rutiinin ohjauksessa olevat moottorikäytöt ovat tavallisesti nopeuden käskyarvoon perustuvia, jossa paikka lasketaan integroimalla nopeuden arvo. Näin rutiinin jokaisen suoritusjakson jälkeen nopeuden ohjauksen käskyarvo säädetään paikan asetusarvon, oloarvon ja muiden säätöparametrien perusteella. Jos moottorikäyttö on paikkaan perustuvassa ohjaustilassa, takaisinkytketyn säädön silmukkaa ei käytetä ja SVB-rutiinin antamat asetusarvot ovat suoraan käytön lähtöarvoja.[10]

NC-säätimen tarkempi malli näkyy lohkokaaviona kuvassa 35.




Kuva 35. Liikkeen ohjaimen tarkempi lohkokaavio.[10]

## 4.2 Sähkömoottori

Servokäytön sähkömoottorina voi olla dynaamisilta ominaisuuksiltaan hyvä AC- tai DC-moottori. Toisin sanoen servokäyttöön sopii minkälainen sähkömoottori tahansa, jota voidaan ohjata suljetulla takaisinkytkennällä.

Työssä käytetty kestomagnetoitu, 6-napainen DC moottori on tyyppiä AM8113-0F20-0000. Tekniset tiedot näkyvät kuvasta 36.

| Data for 50 V DC | AM8113-wFyz |
|---|---|
| Standstill torque | 0.52 Nm |
| Rated torque | 0.50 Nm |
| Rated speed | 3000 min-1 |
| Rated power | 0.16 kW |
| Peak torque | 2.04 Nm |
| Standstill current | 4.8 A |
| Peak current | 18,0 A |
| Torque constant | 0.12 Nm/A |
| Voltage constant | 7.00 mV/min-1 |
| Number of poles | 6 |
| Rotor moment of inertia | 0.067 kgcm² |
| Weight | 0.82 kg |
| Holding torque brake ($M_{br}$) | 0.6 Nm |
| Power consumption (brake) at 24 V DC ($P_{br}$) | 10 W |
| Rotor moment of inertia incl. brake (J) | 0.090 kgcm² |
| Weight incl. brake (m) | 1.02 kg |
| EtherCAT Terminal | EL7211-0010 |

Kuva 36. Moottorin tekniset tiedot.

## 4.3 Anturit

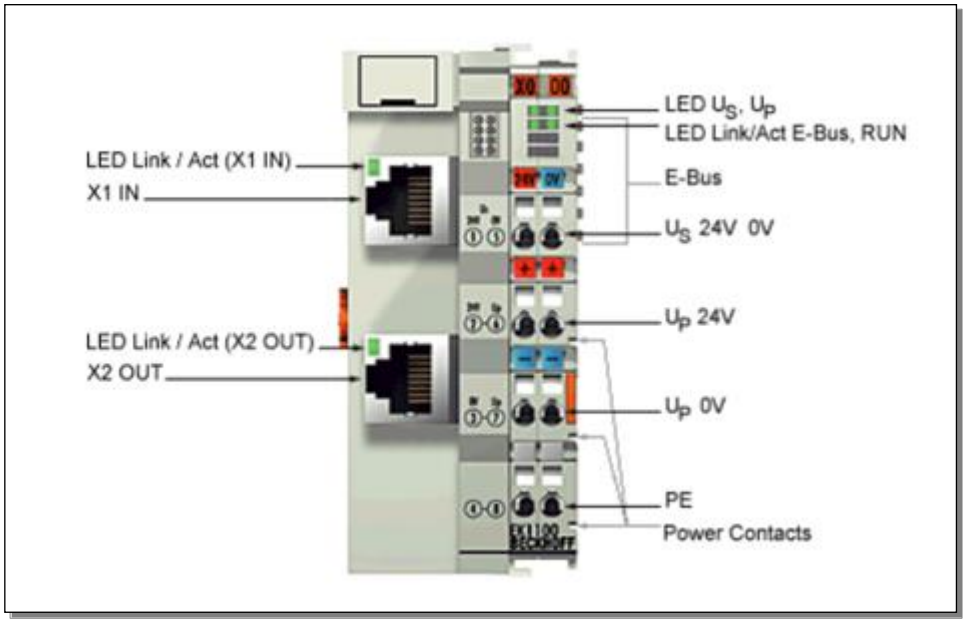
Aistielimistä paikkaa voidaan lukea pulssianturilla ja nopeutta pyörivillä takometrillä. Pulssianturi/enkooderi on elektro-mekaaninen laite, joka muuntaa moottorin akselin kääntökulman tai liikkeen analogiseen tai digitaaliseen koodimuotoon. Niitä on karkeasti jaettuna kahta päätyyppiä, absoluuttisia tai inkrementaalisia. Absoluuttiset enkooderit muistavat paikkansa käyttöjännitteen katkeamisen jälkeen. Ne ovat valmistusvaiheessa sovitettu ohjattavien laitteistojen kanssa, joten enkooderin arvo ja fyysinen paikka ovat aina tiedossa.

Inkrementaalisessa enkoodereissa paikka ei säily muistissa ja se täytyy kotouttaa tunnettuun referenssipisteeseen.

Työssä käytössä olleessa moottorissa (AM8113-0F20-0000) oli integroitu absoluuttinen enkooderi.

## 4.4 Ethercat-automaationväylä ja pääteaste EK1100

Kuvassa 37 näkyy Ethercat master -moduuli, johon tulee liitäntä myös I/O-moduuleille.



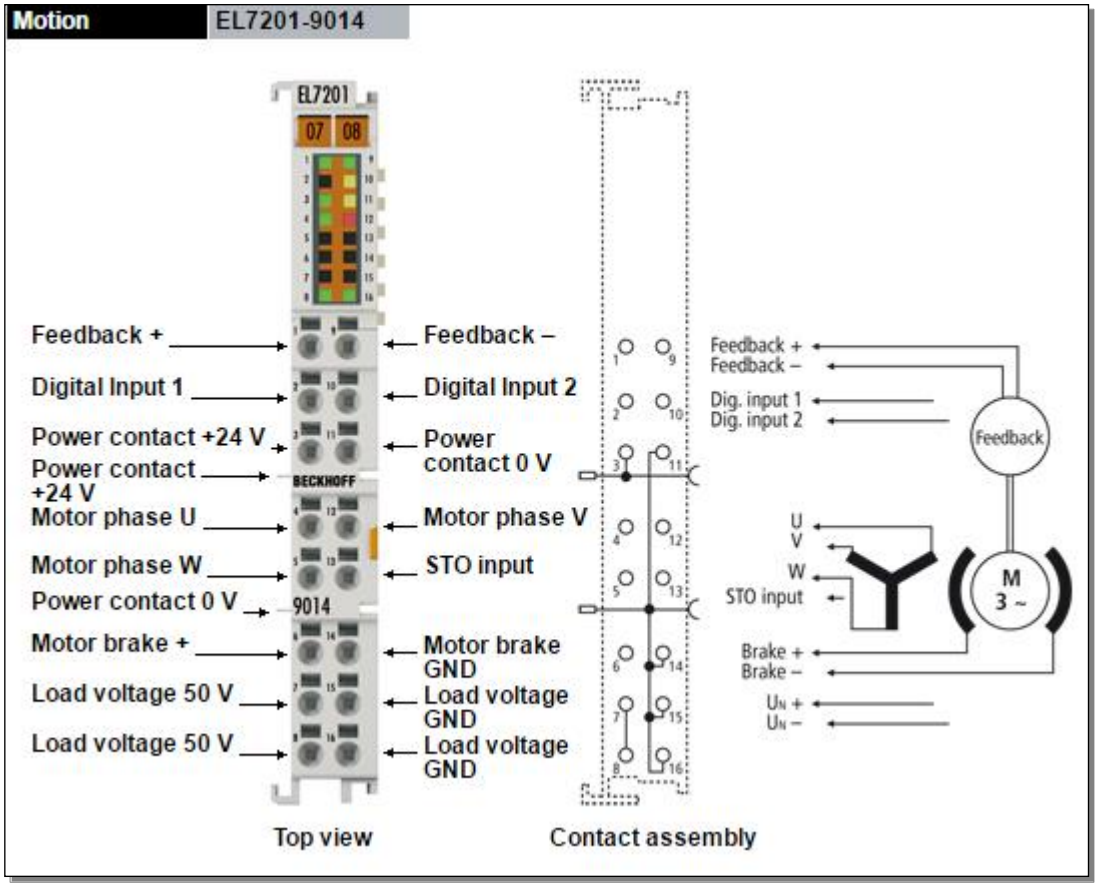

Kuva 37. Ethercat-väylän pääteaste EK1100.

EK1100-kytkin liittää Ethercat-väylään ELxxxx-sarjan I/O-moduulit. Yksi asema pitää sisällään EK1100-kytkimen, mielivaltaisen määrän I/O-kytkentäpisteitä ja väylän pääteasteen. Kytkin muuntaa Ethernetin 100BASE-TX-sähkesanomat E-väylän signaalimuotoon.

Kytkin on liitetty verkostoon ylemmän Ethernet-rajapinnan kautta. Alempaa RJ45-liitintä voidaan käyttää Ethercat-verkoston laajentamiseen ketjuttamalla siihen muita Ethercat-laitteita.

## 4.5 Servovahvistin yksikkö EL7201

Kuvassa 38. näkyy työssä käytetty servovahvistin moduuli EL7201.




Kuva 38. Työn servovahvistin moduuli EL7201.

EL7201-9014 on servomoottorivahvistin pääteaste Ethercat-väylään. Se pitää sisällään integroidut paikannuksen liitännät takaisinkytkennälle. Vahvistin saa virtansa kenttäväylästä ja takaisinsäätö on PI-tyyppiä. Näin paikannus on tarkkaa ja nopeaa. Valvontaparametreja on useita, mm. ylijännite, alijännite, oikosulkuvirta ja moduulin lämpötila. Moottorin kuormitus lasketaan I2T-mallilla. Ethercat-väylä tarjoaa järjestelmänviestinnän ja (”CAN-over-EtherCAT (CoE)”) protokolla on sovelluskerroksena PC-pohjaiselle ohjausteknologialle.[5]

Kuvan 39. taulukosta vahvistimen olennaisin tieto on jatkuvan virran tehollisarvo, 2,8 A.

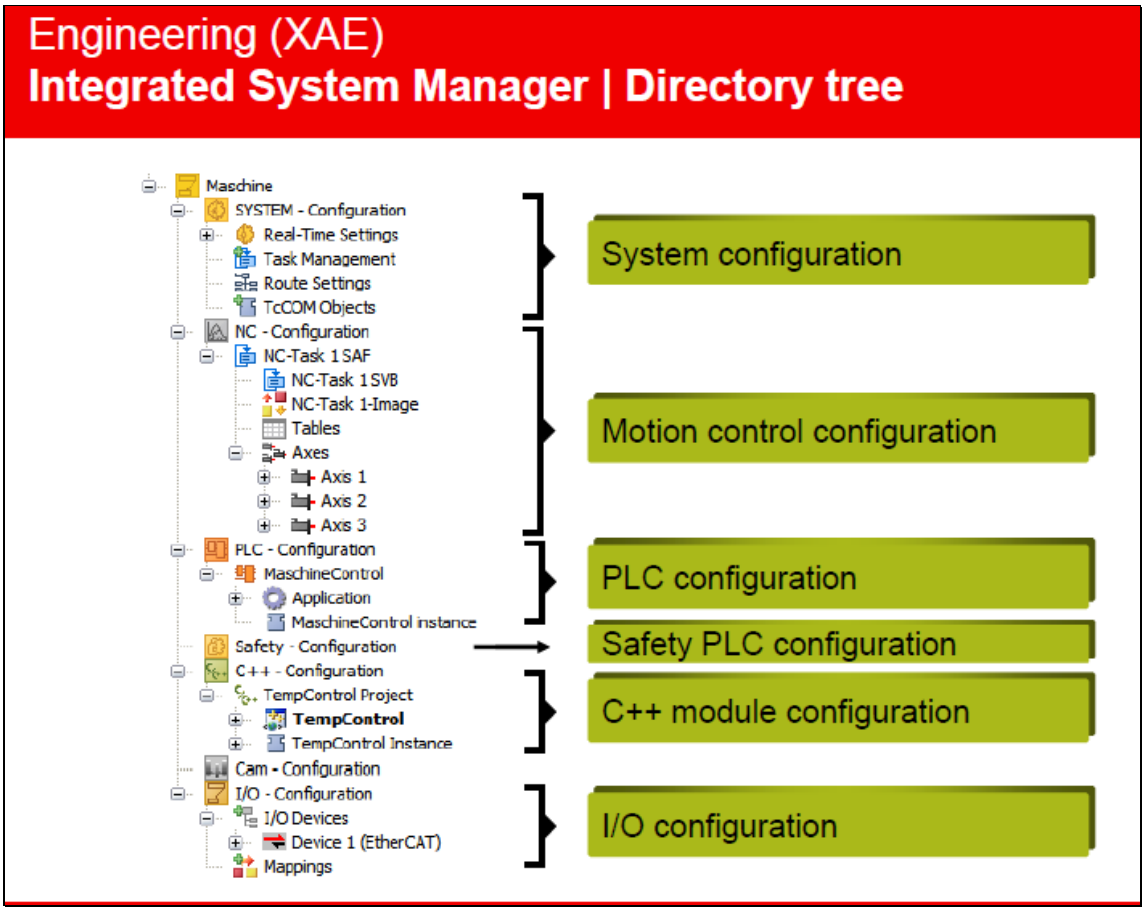
| Technical data | EL7201-9014 |
|---|---|
| Number of channels | 1 servomotor, absolute feedback, motor brake, 2 digital inputs, 1 STO input |
| Connection method | direct motor connection |
| Load type | permanent-magnet synchronous motors |
| Nominal voltage | 8…50 V DC |
| Output current $I_N$ | 2.8 A (rms) |
| Peak current $I_N$ | 5.7 A (rms) for 1 s |
| Frequency range | 0…599 Hz |
| PWM clock frequency | 16 kHz |
| Current controller frequency | 32 kHz |
| Rated speed controller frequency | 16 kHz |
| Output voltage motor brake | 24 V DC (+6 %/-10 %) |
| Output current motor brake | max. 0.5 A |
| Current consumption power contacts | typ. 50 mA + holding current motor brake |
| Current consumption E-bus | 120 mA |
| Special features | compact (only 12 mm wide), system-integrated, absolute feedback, One Cable Technology (OCT), plug-and-play, STO (Safe Torque Off) |
| Weight | approx. 60 g |
| Operating/storage temperature | 0…+55 °C/-25…+85 °C |
| Relative humidity | 95 %, no condensation |

Kuva 39. Servovahvistimen tekniset tiedot.[5]

# 5 TwinCAT SYSTEM MANAGER -järjestelmänhallinta ja laitteiston määritykset

TwinCAT System Manager on tärkein työkalu Twincat-järjestelmän määrittelyssä. Siinä muodostetaan ohjelmallisten tulo- ja lähtörutiinien sekä kenttäväylän fyysisten tulo- ja lähtöliitäntöjen väliset yhteydet. Loogisten tulojen ja lähtöjen sijoitetaan fyysisiin tuloihin ja lähtöihin yhdistämällä ohjelmarutiinien muuttujat kenttäväylän muuttujiin.

Kuvassa 40. Näkyy järjestelmähallinnan hakemistopuu, joka kuuteen pääosaan. Ylinpänä on järjestelmän asetukset (”System configuration”). Seuraavaksi tulevat liikkeen ohjauksen määritykset (”Motion Control”). Tämän jälkeen tulevat PLC-osion määritykset, turva-logiikan osio, C++-moduulin määritykset, sekä viimeisenä I/O-osio.




Kuva 40. Järjestelmähallinnan hakemistopuu.

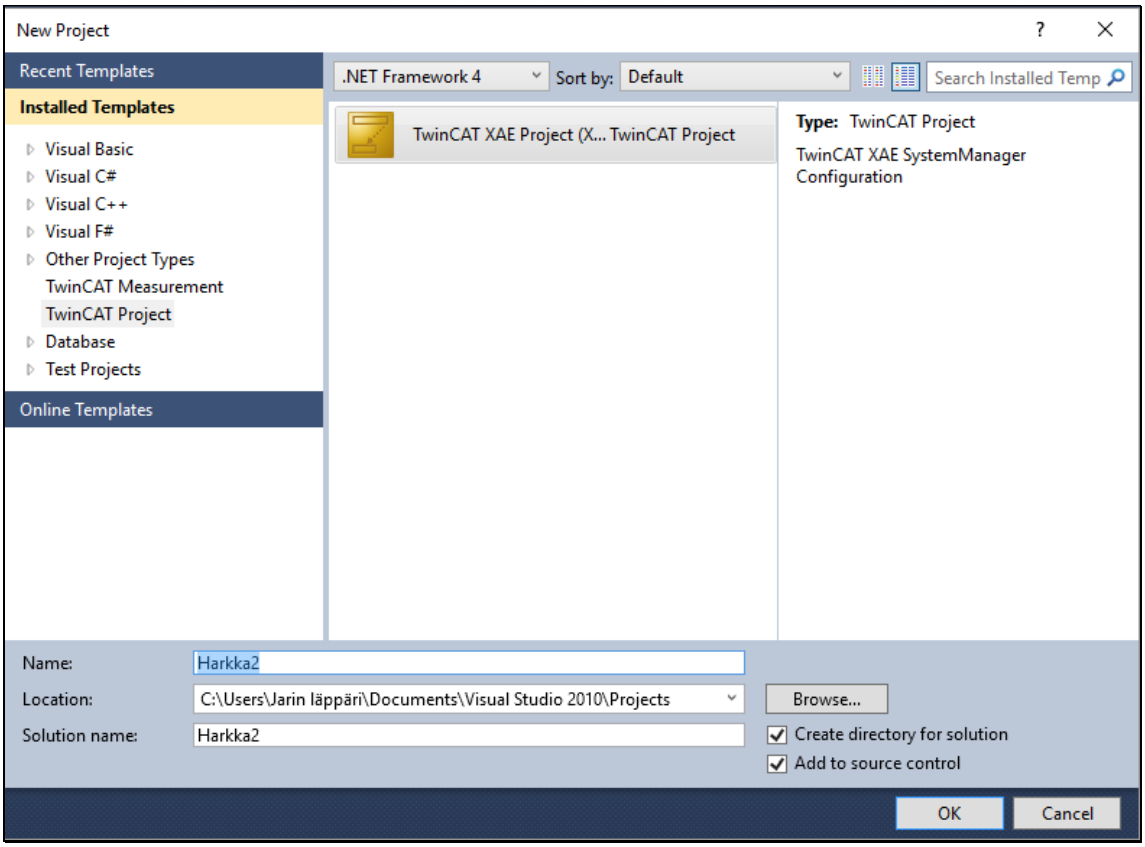
TwinCAT System Manager on muuttuja-orientoitunut. Tämä tarkoittaa sitä, että pienin tietotyyppi, jolle voidaan antaa osoite ja yhdistää on muuttuja. Muuttujan tietotyyppi voi olla yksittäinen bitti, tavu, 16-bittinen sana, 32-bittinen sana tms. Muuttujat voivat olla tyypiltään myös struktuureja, taulukoita tai kenttiä, jotka sisältävät muita tietotyyppejä.

### 5.1.1 Prosessikuvat

Prosessikuvat ovat toimilaitekohtaisia luetteloita tulo- ja lähtömuuttujista. Jokainen toimilaite sisältää oman prosessikuvansa, jossa määritellään muuttujat. Esim. kenttäväylällä, I/O-laitteilla tai NC-liikkeen ohjauksella on omansa. Poikkeuksena on PLC-ohjelman ja reaaliaika ohjauksen järjestelmä-rutiinien yhteinen prosessikuva.

## 5.2 Uuden PLC projektin luominen

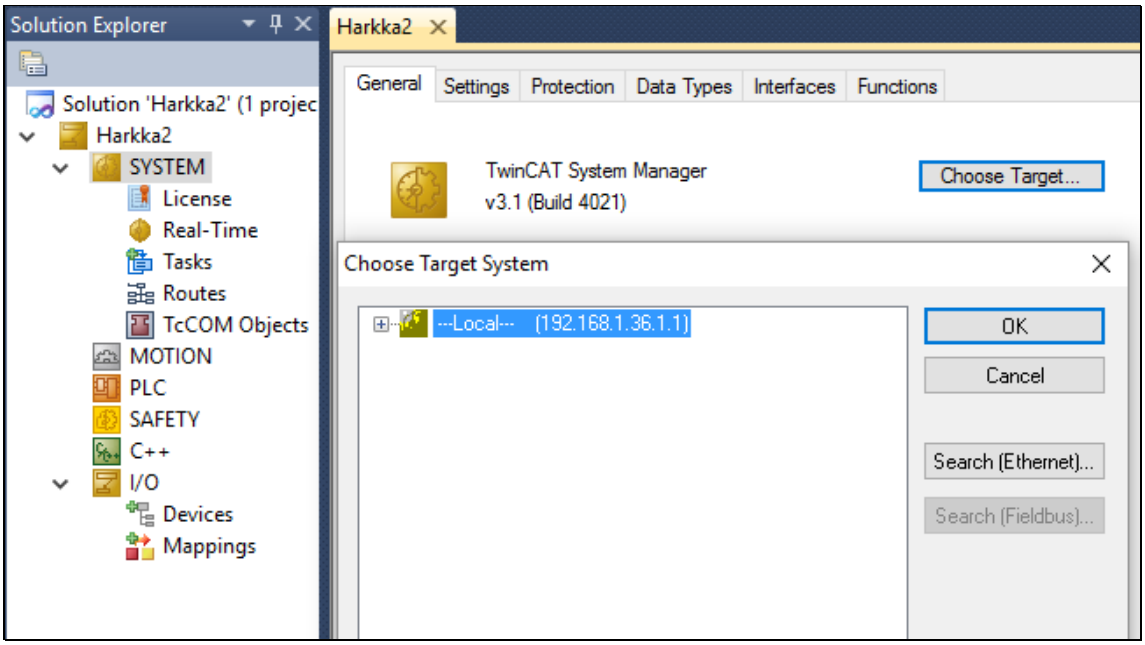
PLC-projektin luominen alkaa käynnistämällä Microsoft Visual Studio 2010 -kehitysympäristö ja valitsemalla projektipohja (”Template”) kohdasta ”New Project”->”TwinCAT Project”->”TwinCAT XAE Project (XML format)”. Projekti nimetään ja hyväksytään OK-painikkeella kuvan 41. mukaisesti. Luotu projekti oli XML-tiedostomuodossa.




Kuva 41. TwinCAT (XAE) -projektin luominen.

Luotu projekti näkyy järjestelmähallinnan ”Solution Explorer” -projektipuussa, jossa projektin rakenne jäsentyy eri ohjelmamoduuleihin. Nämä luodun projektin jäsenet/ohjelmamoduulit näkyvät luettelona projektiprofiilin alapuolella ollen tässä tapauksessa: SYSTEM, MOTION, PLC, SAFETY, C++ sekä I/O.

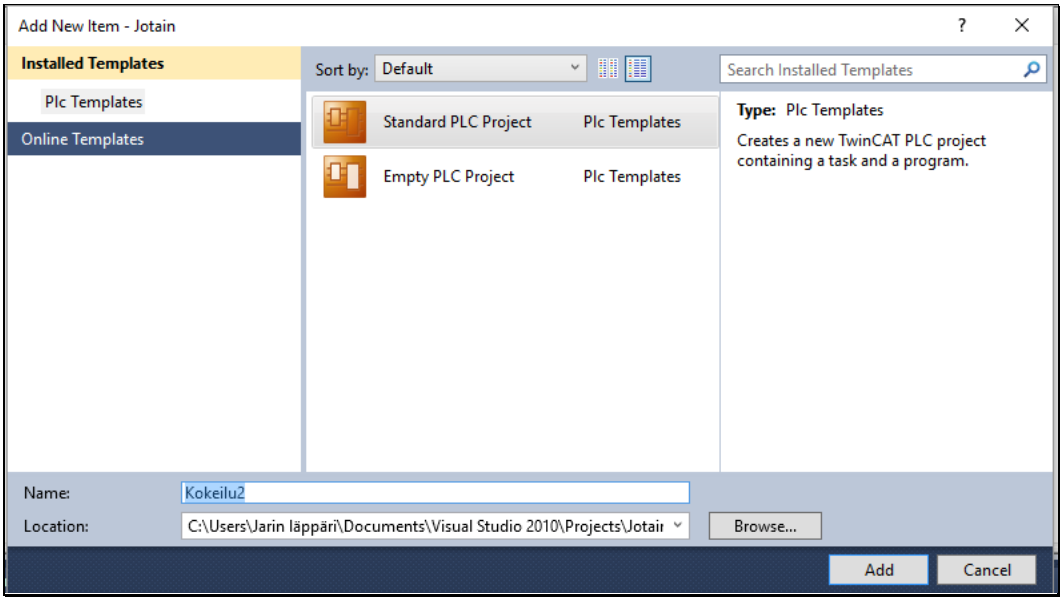
Seuraavaksi määritellään käytön projektin asennustapa paikalliseksi (”local”). Tämä tapahtuu kuvan 42. mukaisesti projektipuun SYSTEM-solmun kohdassa General->”Choose Target…”. Valitaan ehdotettu ip-osoite 192.168.1.36.1.1 ja kuitataan OK-painikkeella.




Kuva 42. Järjestelmän määritteleminen paikalliseksi yksiköksi.

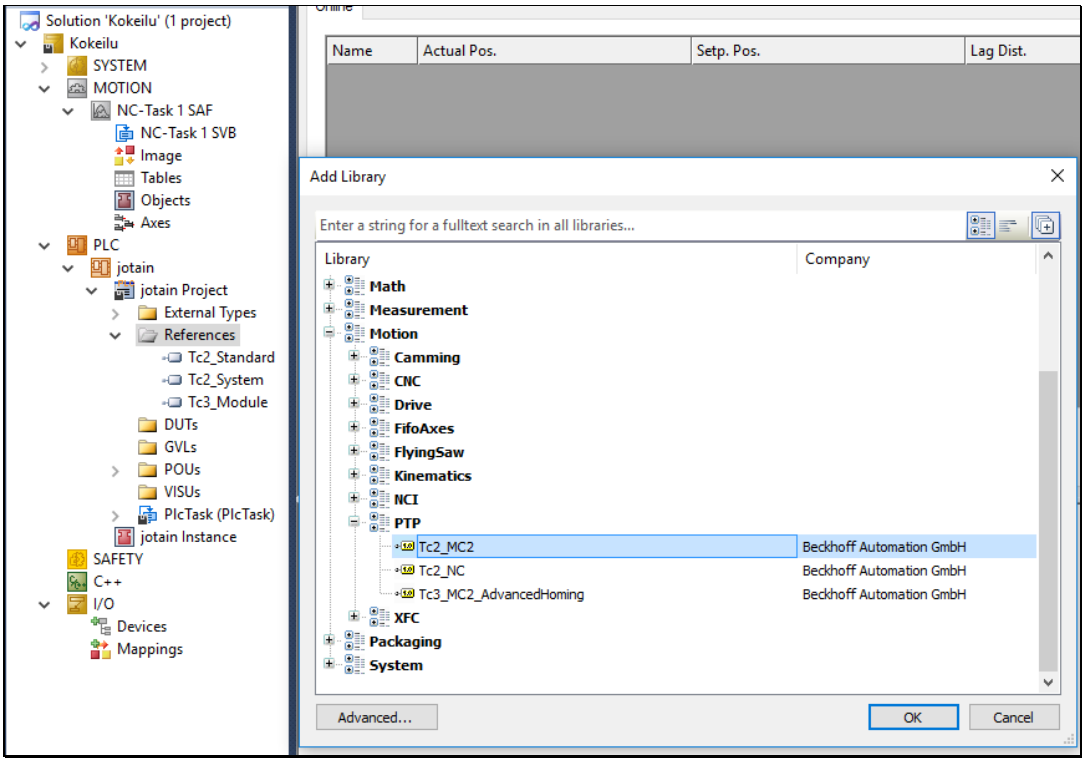
### 5.2.1 PLC-ohjelmamoduulin määritykset

PLC-ohjelman luonti alkaa solmun kohdan pudotusvalikosta ”Add New Item”, josta aukeaa kuvan 43. kaltainen näkymä PLC-projektimalleihin (”templates”). Valitaan ”Standard PLC Project”, annetaan projektille nimi ja kuitataan painikkeesta ”Add”.



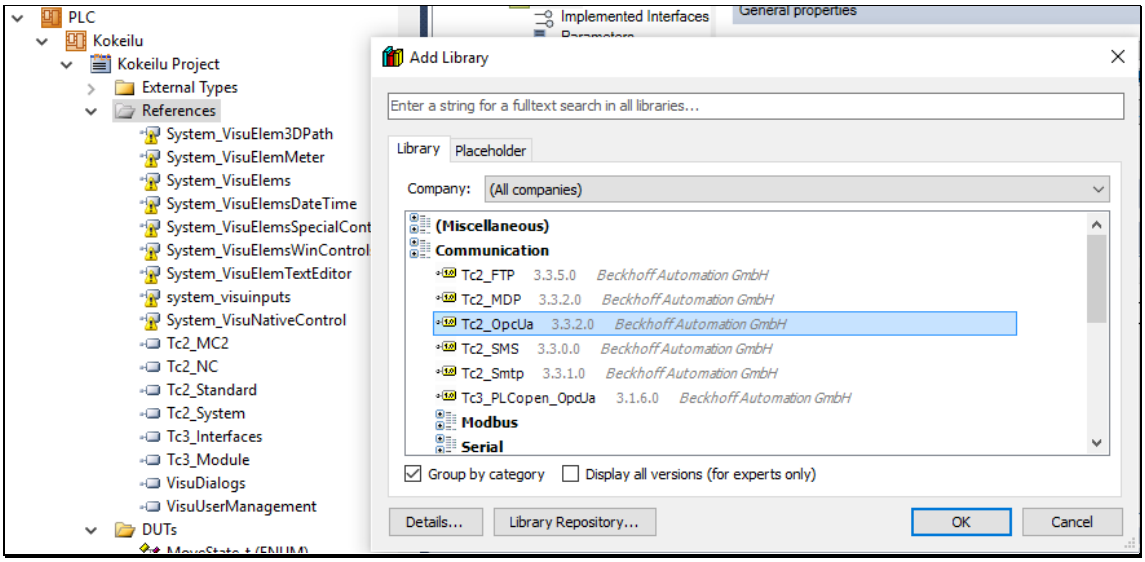

Kuva 43. PLC projektin luominen.

Luotuun PLC projektiin täytyy lisätä moottorikäytön ja liikkeen ohjauksen kirjastot. Tämä tapahtuun kuvassa 44. PLC-solmun alihakemiston kohdasta ”Reference”->”Add Library”, josta valitaan kirjastot ”TC_MC2” ja ”TC_NC”.




Kuva 44. Moottorin ja liikkeen ohjaimen kirjastojen lisääminen PLC-projektiin.

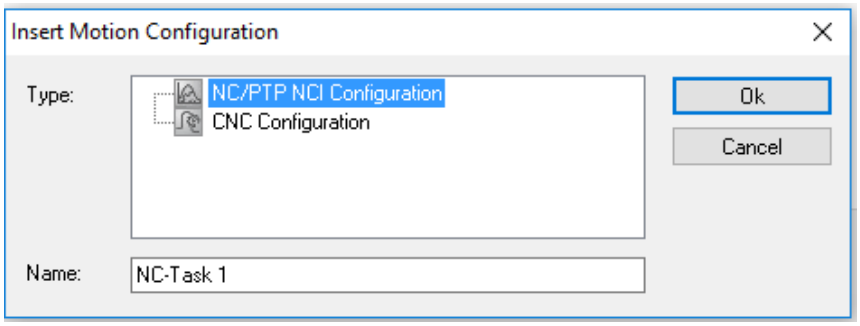
Projektiin lisättiin myös OPC_UA-kirjastot kuvan 45. mukaisesti ulkoista kommunikaatiota ajatellen.




Kuva 45. OPC_UA-kirjastojen lisääminen PLC-projektiin.

## 5.3 NC-liikkeen ohjaimen rutiinin luominen ja määritykset

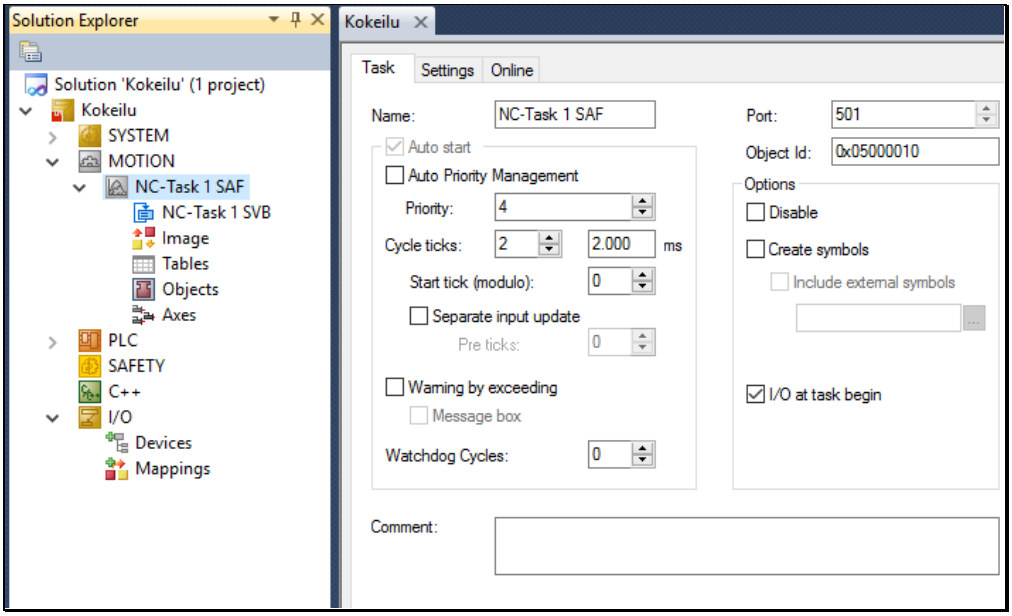
Kuvan 46. mukaisesti ”Solution Explorer” projektipuun kohdasta ”MOTION” tulee pudotusvalikko “Insert Motion Configuration”. Valitaan “NC/PTP NCI Configuration” vaihtoehto, jolloin NC-rutiinin nimeksi tulee järjestelmän antama ”NC-Task 1”. ”Solution Explorer” projektipuun alapuolelle ilmaantuu luotu profiili ”NC-Task 1”.




Kuva 46. Liikkeen ohjaimen rutiinin luominen ja lisääminen projektiin.

Järjestelmän hallinnassa luotu profiili ”NC Task 1” näkyy ”Task” -välilehdellä ”NC Task 1 SAF”, joka takoittaa liikkeen ohjaimen takaisinkytkennän (SAF) rutiinin asetuksia.

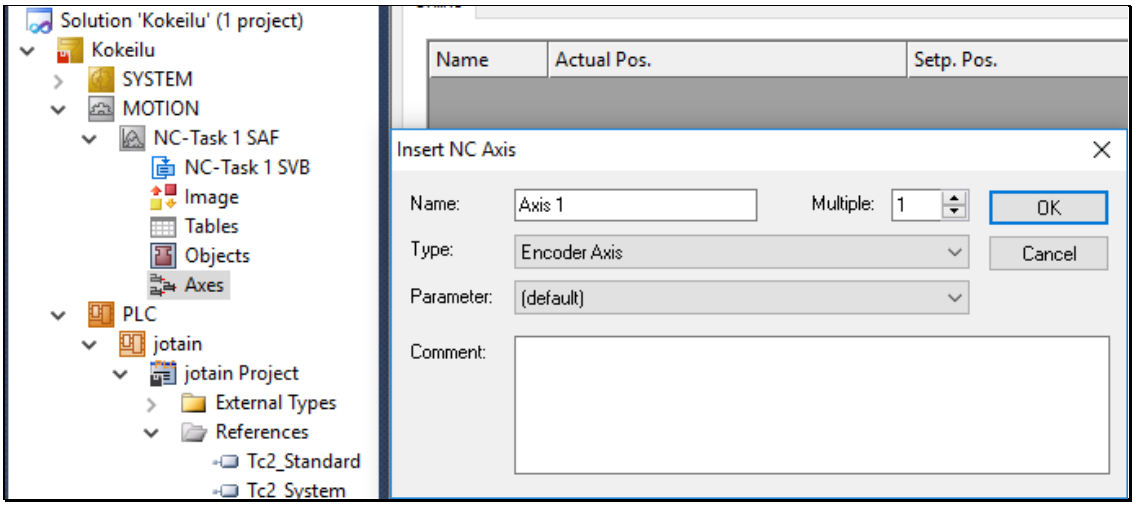
Säätöarvoista takaisinkytkennän rutiinille annettiin 2 ms:n aika-ikkuna. Muita asetuksia ovat ADS-kommunikaation liikkeen ohjaimen porttin parametrin arvo 501.




Kuva 47. Liikkeen ohjaimen takaisinkytkennän (SAF) rutiinin asetukset.

### 5.3.1 Käytön akseliprofiilin lisääminen NC-liikkeen ohjaimeen

”Solution Explorer” hakemistopuussa on luettelo kaikista Twincatin moduuleista. Kuvan 48. mukaisesti akselin lisääminen kokoonpanoon tapahtuu kohdassa ”MOTION”->”Axes”->”Add new.item”->”Insert NC axis”. Valitaan tyypiksi: ”Encoder Axis”. Järjestelmä nimeää akselin ”Axis 1”: i ja tämä on luodun NC-akseliprofiilin objekti.

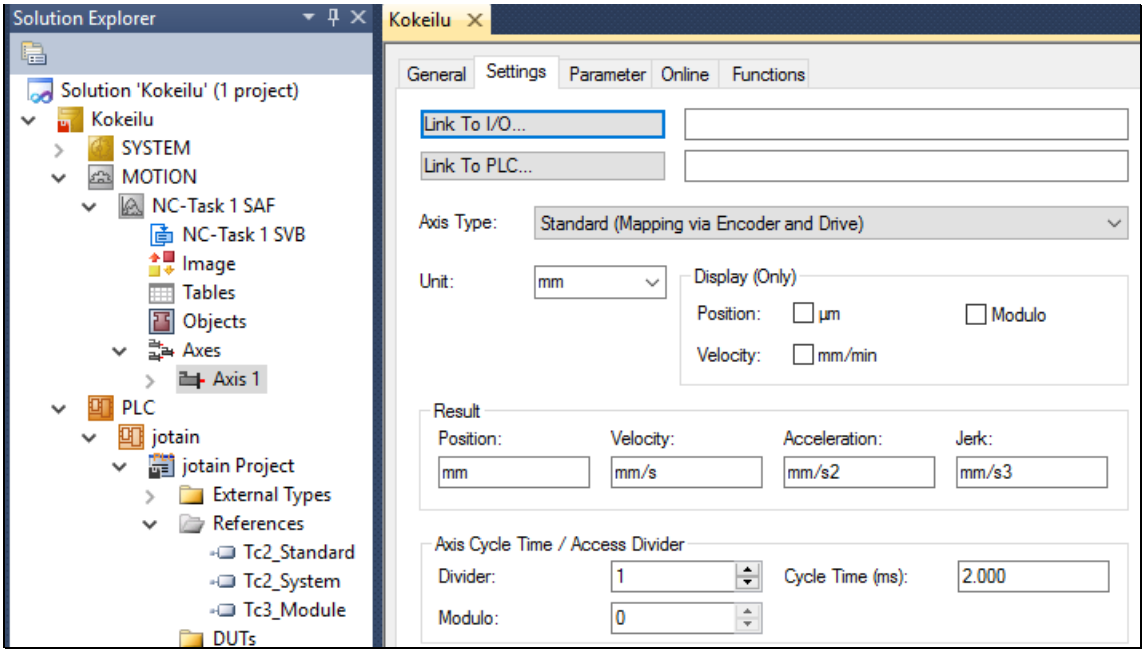



Kuva 48. Käytön akseliprofiilin lisääminen NC-liikkeen ohjaimeen

### 5.3.2 NC-Akselin yleiset asetukset

NC-akselin asetuksissa määritellään useita moottorin ohjauksen ja takaisinkytkennän parametreja, kuten enkooderin skaalauskerroin, kiihtyvyyden, hidastuvuuden ja hetkittäisen kiihtyvyyden arvot.

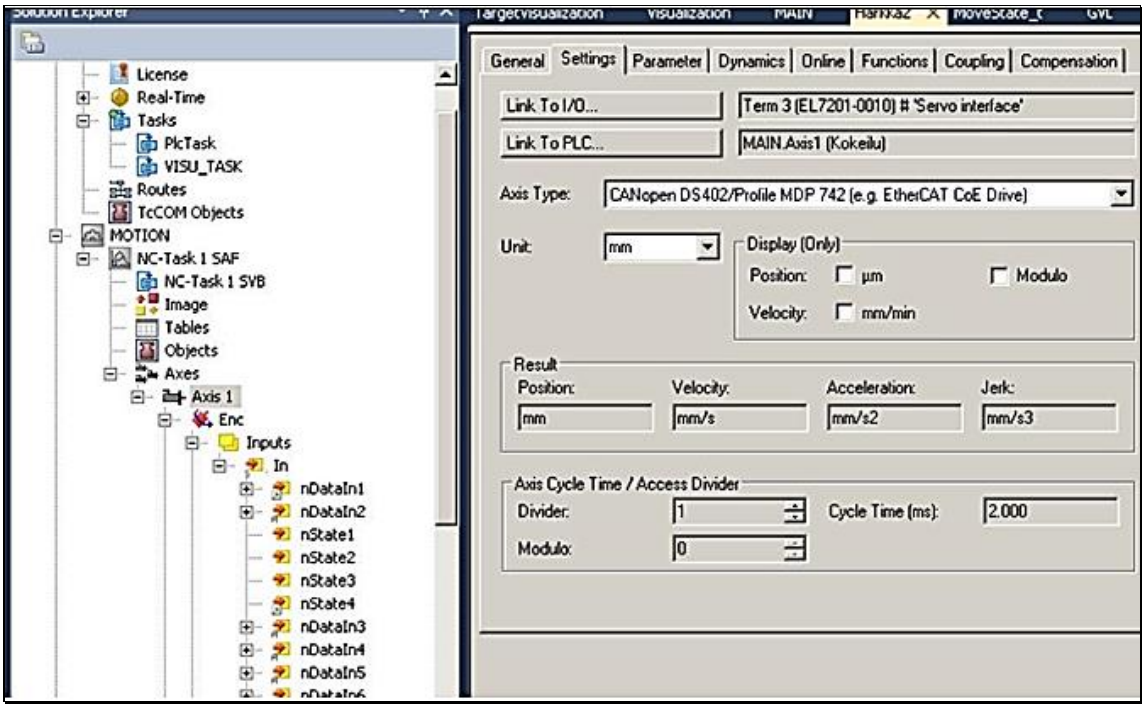
Kuvassa 49. on akseliprofiilin ”Axis 1”:n näkymä välilehdeltä “Settings”.




Kuva 49. Näkymä käytön akseliprofiilin asetusten välilehdelle.

### 5.3.3 NC-akselin ja PLC-ohjelman yhdistäminen

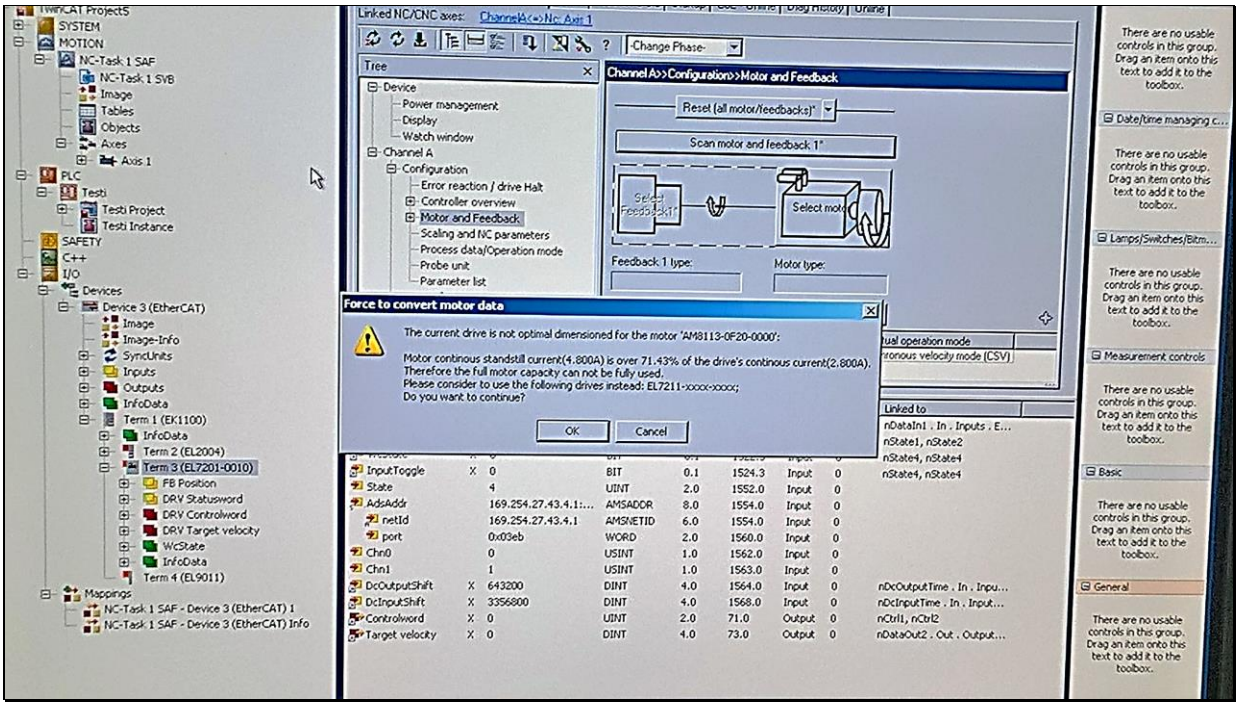
Kuvan 50. hakemistopuun kohdassa ”Axis1”->”Settings” välilehdeltä yhdistetään I/O:n servovahvistin PLC:n NC-akseliin ”Axis 1”.




Kuva 50. NC-akselin yhdistäminen PLC-ohjelmaan sekä I/O:n liittäminen projektiin.

## 5.4 I/O laitteiden määritykset

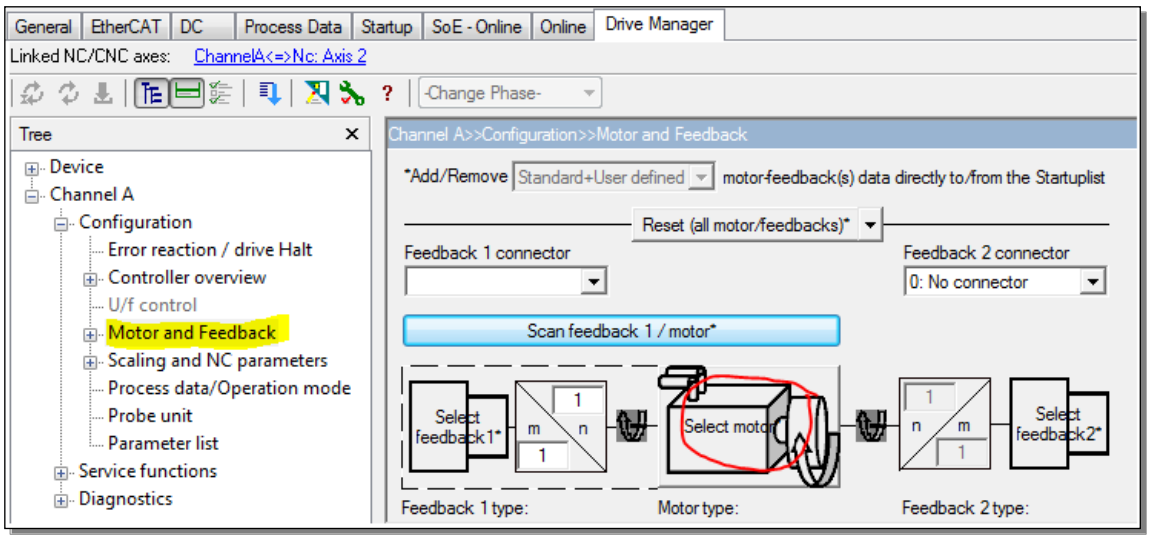
Järjestelmähallinnassa I/O laitteiden liittäminen projektiin on hoidettu automaattisen, ”Scan for Boxes” -liitäntätoiminnon avulla. Järjestelmä havaitsi liitetyn Ethercat-väylän pääteasteen, EK1100:n ja loi sille projektipuuhun solmun ”Term 1”. Vastaavasti servo-vahvistimelle luotiin projektipuuhun solmu ”Term 3”.




Kuva 51 I/O laitteiston, servovahvistimen liittäminen projektiin.

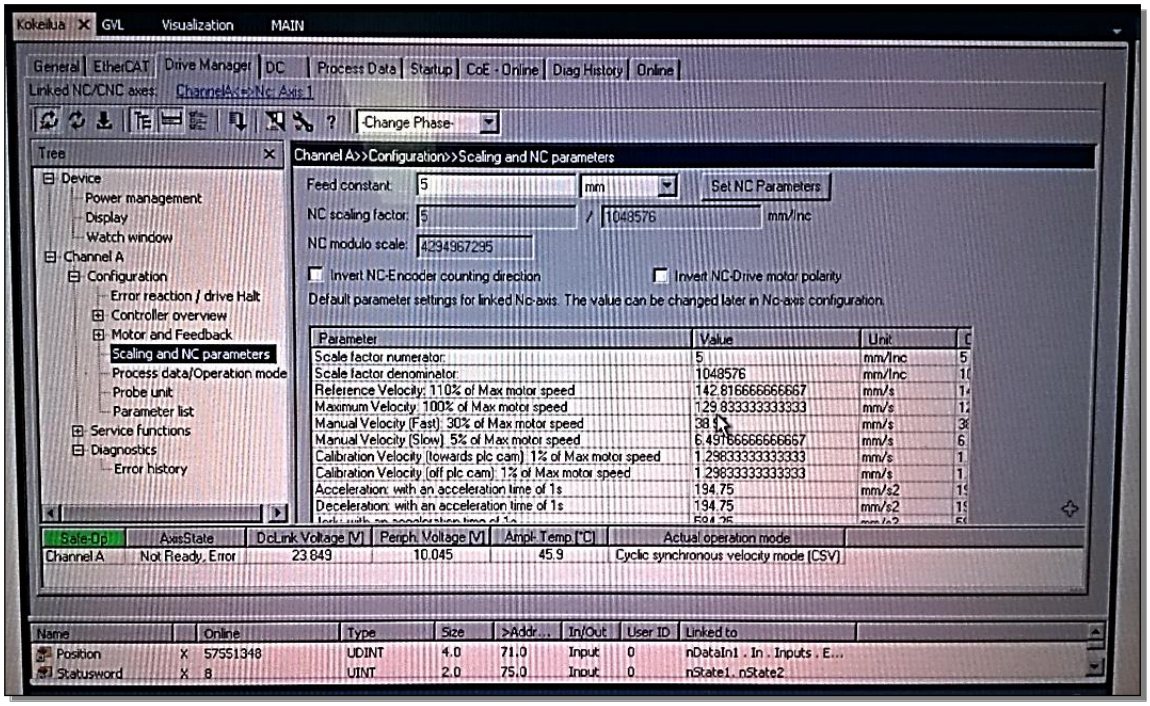
### 5.4.1 Moottorin valinta

Kuvan 52. moottorin valinta onnistuu automaattisesti “Scan feedback 1/motor” napilla. Käytettäessä Bechoffin omaa moottoria löytyi sille ja takaisinkytketylle enkooderille valmiit mallit.




Kuva 52. Moottorin valinta ja asetukset.[8]

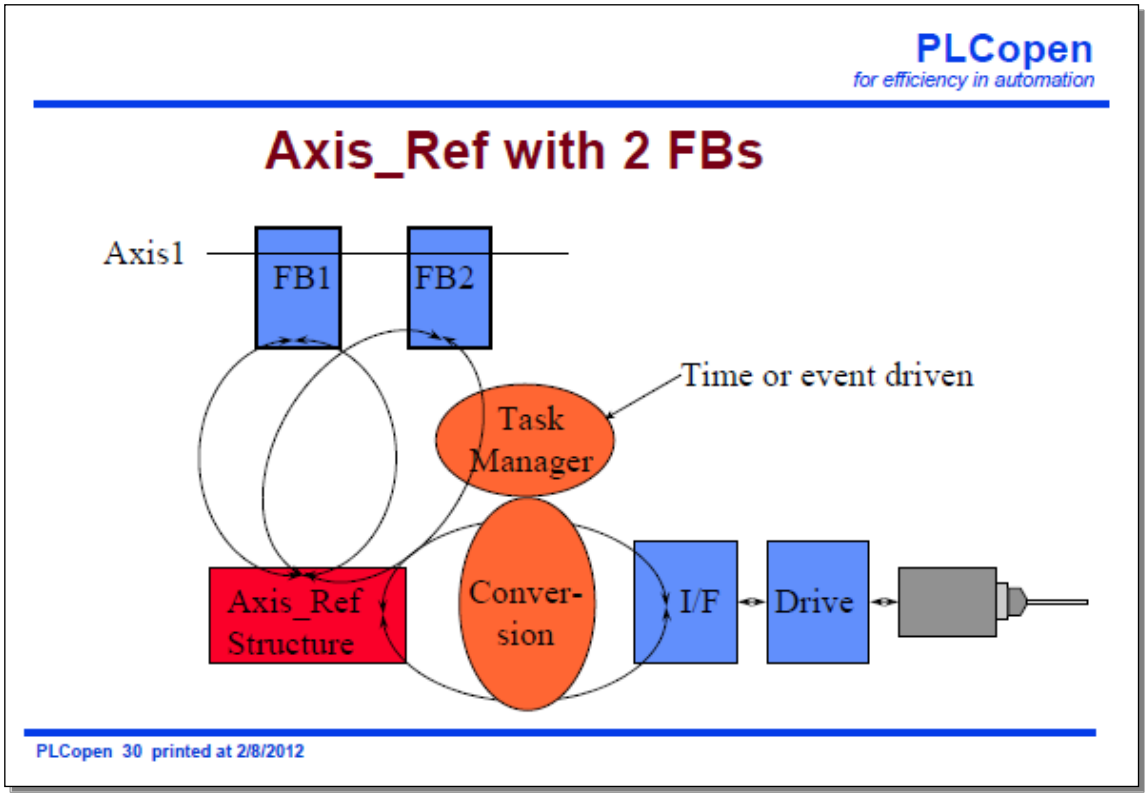
Kuvan 53 asetuksista “Scaling and NC parameter” säädetään takaisinkytkennän ”Feed Constant” -parametria. Tämä syöttöparametri määrittää pulssianturin kulkeman ympyrän kehän pituuden/matkan moottorin pyörähtäessä yhden kierroksen. Työssä käytetty asetus oli 5 mm/Inc.




Kuva 53. Moottorin takaisinkytkennän syöttöparametrin asettaminen arvoon 5 mm/Inc.

# 6 PLC-ohjelmointi

Liikkeenohjauksen toimintamalli Beckhoffin Twincat 3-ympäristössä on ”PLCopen” standardin mukaista. Toimintamallin yleisluontoinen rakenne on nähtävissä kuvasta 54.




Kuva 54. ”PLCOpen”-standardin mukainen liikkeenohjauksen malli. [18]

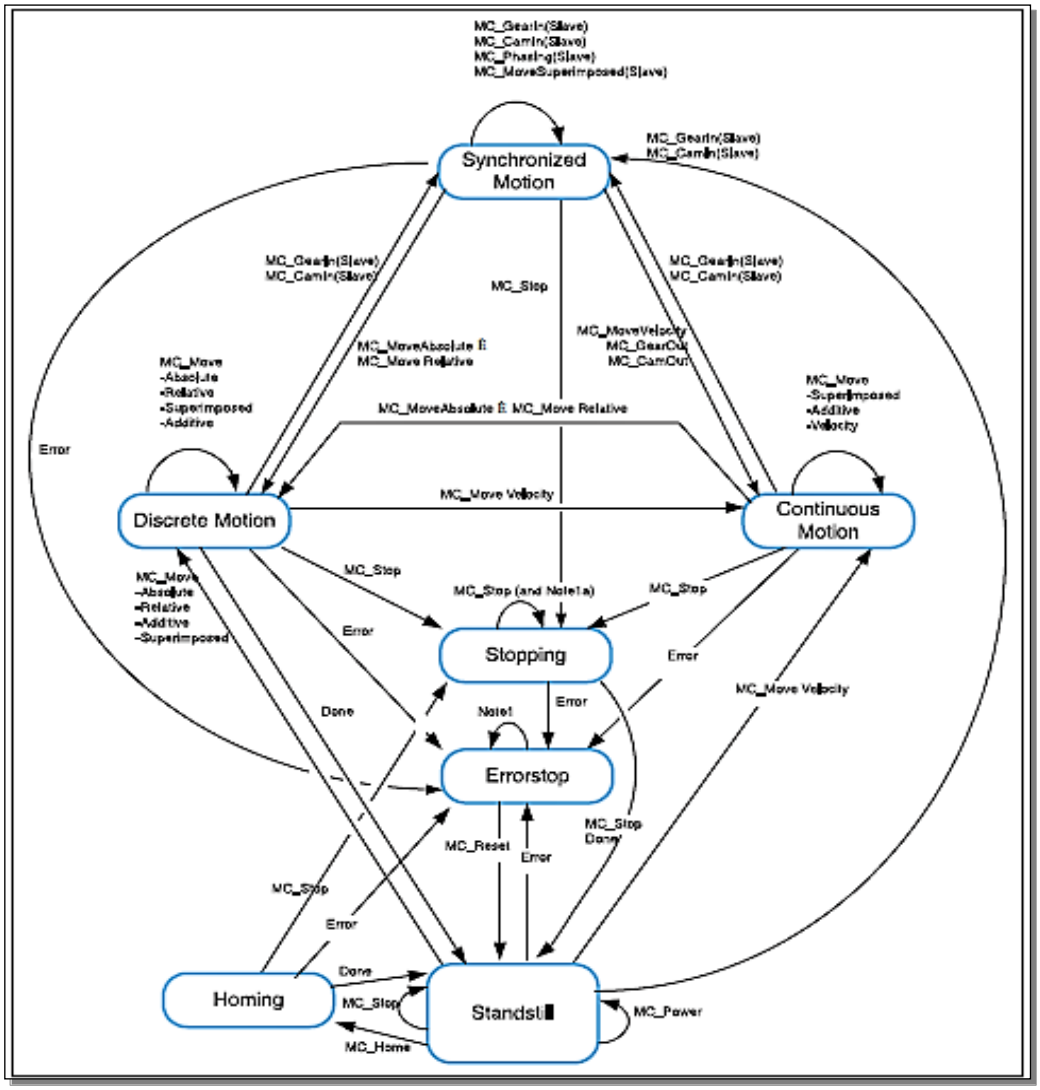
Liikkeen ohjaimen(NC) funktioblokit ovat FB1 ja FB2 ja tallentavat tilansa Axis_Ref-struktuurin muuttujaan Axis 1.

Järjestelmänhallinnan rutiinien asetukset ja ohjaus näkyy kuvassa ”Task Manager” nimikkeenä. ”Task Manager”:in ohjaus on ajoitus- tai tapahtuma-ohjattua. [18]

## 6.1 ”PLCopen” -standardin määritelmät liiketiloille

”PLCopen”-standardin liikkeen ohjauksen PLC-kirjastot määrittelevät moottorin kinemaattisten liiketilojen ohjaukseen käytettävät funktioblokit (”function blocks”). Moottorin pyörimisakselin liiketilaa tai olosuhdetta kuvataan liikkeen tilakoneella (”state machine”). Tilakone määrittelee mitä funktioblokkeja on sallittu kutsuttavan tietyssä liiketilassa.

Kuvassa 55. näkyy kaikki seitsemän liiketilaa.




Kuva 55. ”PLCOpen”-standardin mukainen tilakone, eli määritelmä liiketiloille. [18]

Tilakoneessa on kuvan 55. mukaisesti seitsemän määriteltyä tilaa:

1. Pysähdyksen tila ("Stand Still").

2. Kotiutus-tila ("Homing").

3. Erillisen liikkeen tila ("Discrete Motion").

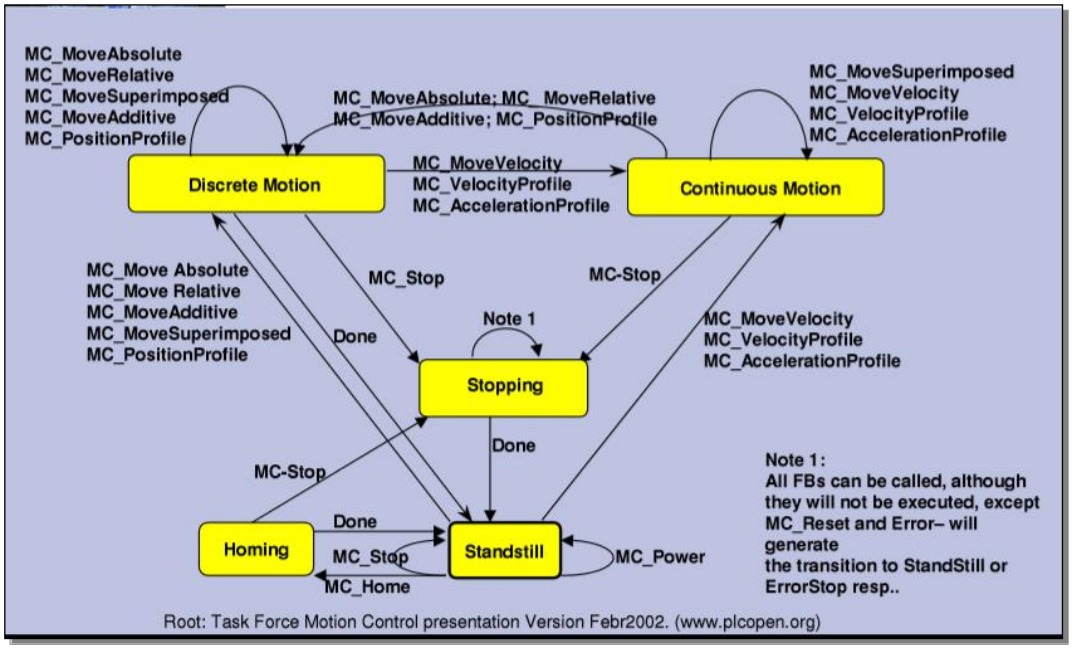
4. Jatkuvan liikkeen tila ("Continuous Motion").

5. Tahdistetun liikkeen tila ("Synchronized Motion").

6. Pysähtyneeseen tilaan siirtyminen ("Stopping").

7. Virhepysähdyksen tila ("Error Stop").

Tämän työn pääasialliset liiketilat voidaan jakaa kuvasta 56. näkyvään erillisen liikkeen tilaan (”Discrete Motion”) ja jatkuvan liikkeen tilaan (”Continuous Motion”).



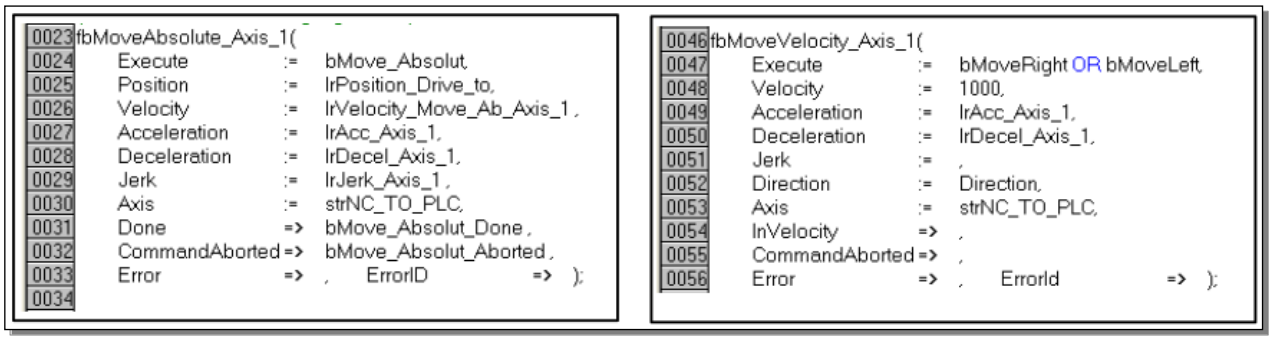

Kuva 56. Liiketilojen jakautuminen erillisen- ja jatkuvan liikkeen tilaan.

Liiketilan kulku lähtee aina pysähtyneen liiketilan kautta (”Standstill”), josta siirtyminen näihin kahteen liiketilaan tapahtuu PLC-ohjelmassa kutsumalla kunkin tilan funktioblokkeja.

Kuvassa 57. nähdään PLC-ohjelmassa käytettyjä ja ST-kielellä määriteltyjä ilmentymiä erillisen liikkeen funktioblokista, ”**MC_MoveAbsolute”,** sekä jatkuvan liikkeen funktioblokista ”**MC_MoveVelocity**”.

ST-kielen funktioblokeissa operaattori ”:=” tarkoittaa tulopuolen parametrin tai arvon liittämistä. Lähtöpuoleen liitetyt arvot tai parametrit ilmaistaan operaattorilla ”=>”.

**DISCRETE MOTION** **CONTINUOUS MOTION**

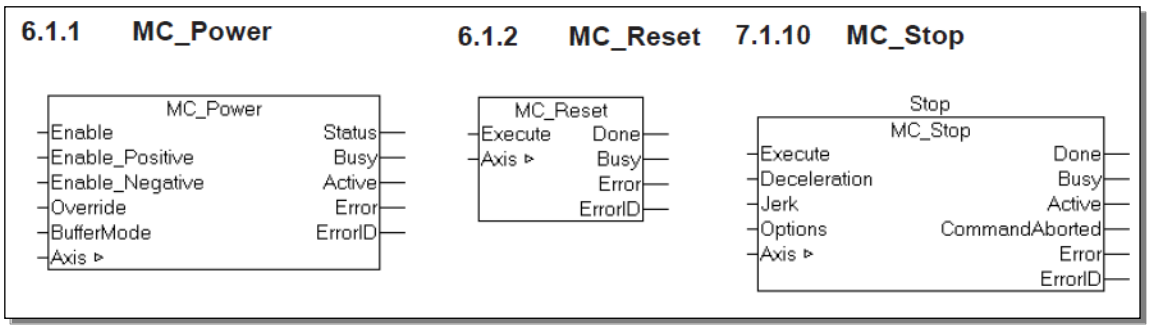



Kuva 57. Ilmentymät ST-kielellä funktioblokeista ” MC_MoveAbsolute” ja ” MC_MoveVelocity”.

Erillinen liiketila perustuu paikkaohjattuun, enkooderilta takaisinkytkennästä saatuun etäisyyden arvoon. Näin palautuminen pysähdyksen tilaan voi tapahtua suorituksen päätyttyä, jolloin funktio blokin lähtöparametri, ”fbMoveAbsolute_Axis1.Done”, saa arvon ”TOSI”. Kuvan 57. ohjelmassa arvo sijoitetaan muuttujaan ”bMove_Absolut_Done”. Vaihtoehtoisesti kesken liiketilan suorituksen voi tulla häiriö, jolloin järjestelmä siirtyy virhepysähdyksen tilaan (”Error Stop”) tai käyttäjä aktivoi ”MC_Stop”-funktioblokin, jolloin tila muuttuu pysähtyneeksi tilaksi.

Jatkuva liiketila on nopeusparametrin mukaan ohjattu päättymätön toiminto. Näin pysäytys tapahtuu ”MC_Stop”-funktioblokin tai palautuu ”Error”-virhetilan myötä pysähtyneeseen tilaan.

Käytön akselin aktivoiva ja pyörimissuunnat vapauttava ns. hallinnollinen funktio blokki on ”MC_POWER”, joka aloittaa ohjelman. Kuvassa 58. näkyy LD-kielellä työssä käytettyjä liikkeen aloitukseen ja lopetukseen liittyviä funktio blokkeja, kuten ”MC_Reset” ja ”MC_Stop”.




Kuva 58. Eräitä työssä käytettyjä funktio blokkeja ovat ”MC_Power”, ”MC_Reset” ja ”MC_Stop”.

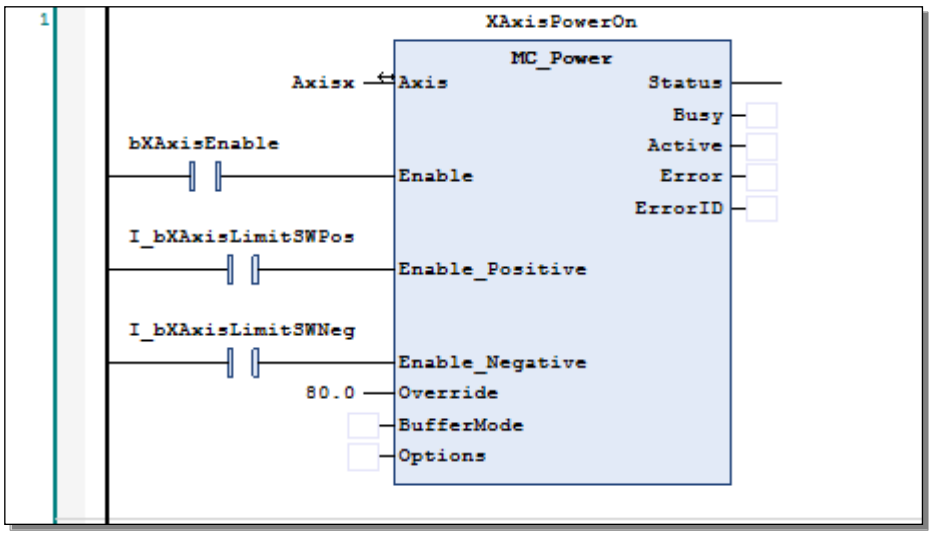
## 6.2 Ohjelman rakenne ja eteneminen

Ohjelman rakenne voidaan jakaa seuraavan luettelon mukaisesti.

1. Käytön akselin aktivoiminen ja vapauttaminen "MC_Power"-funktioblokilla
2. Käytön jatkuvan liiketilan manuaalinen operointi "MC_Jog"-funktioblokilla
3. Käytön akselin kotiuttaminen "MC_Home"-funktioblokilla
4. Käytön akselin paikka-ohjattu operointi "MC_MoveAbsolute"-funktioblokilla
5. Käytön akselin operoiminen paikka-ohjatulla funktioblokilla "MC_MoveAbsolute", jossa on paikalle annettu alku- ja loppuarvo.

### 6.2.1 Käytön aktivointi

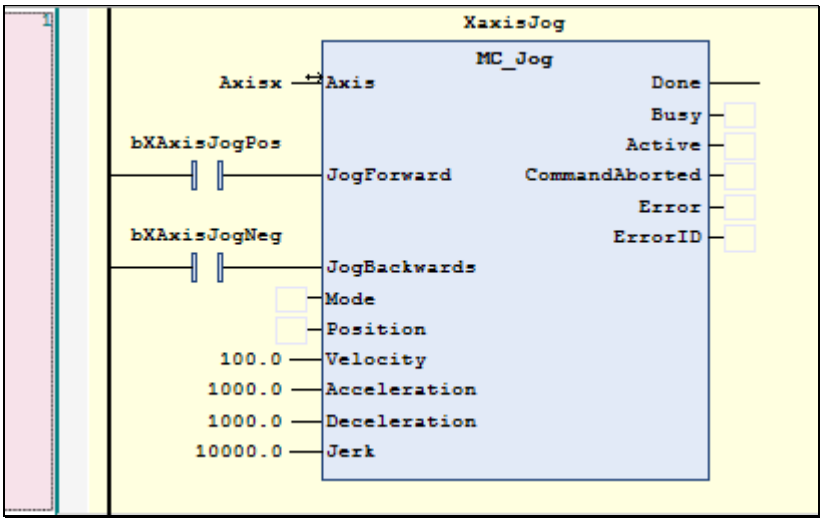
Käyttö aktivoidaan muuttujalla ”bXAxisEnable”, joka on liitetty käyttöliittymän kytkimeen ”X-Axis Enable”. Moottorin pyörimissuunnat vapautetaan muuttujilla ”I_bXaxisLimitSWPos” ja ” I_bXaxisLimitSWNeg”. Näihin muuttujiin liittyvät vastineet käyttöliittymässä ovat ”X-Axis Limit+” ja ” X-Axis Limit-”.




Kuva 59. Käytön akselin ja pyörimissuuntien aktivoiminen ”MC_Power”-funktioblokilla.

### 6.2.2 Käytön manuaalinen operointi

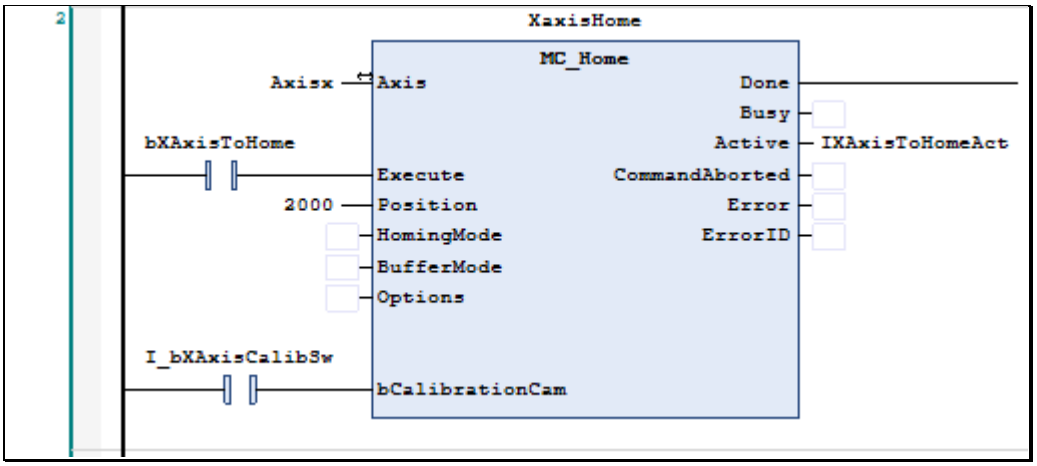
Käytön manuaalinen ajaminen tapahtuu painamalla käyttöliittymän ”X-Axis Jog +” ja ”X-Axis Jog -” -nappeja. Nämä ovat kytketty ”MC_Jog”-funktioblokin tuloihin ”JogForward” ja ”JogBackwards”.[14]




Kuva 60. Käytön manuaalinen operointi ”MC_Jog”-funktioblokilla. [14]

### 6.2.3 Käytön akselin kotouttaminen

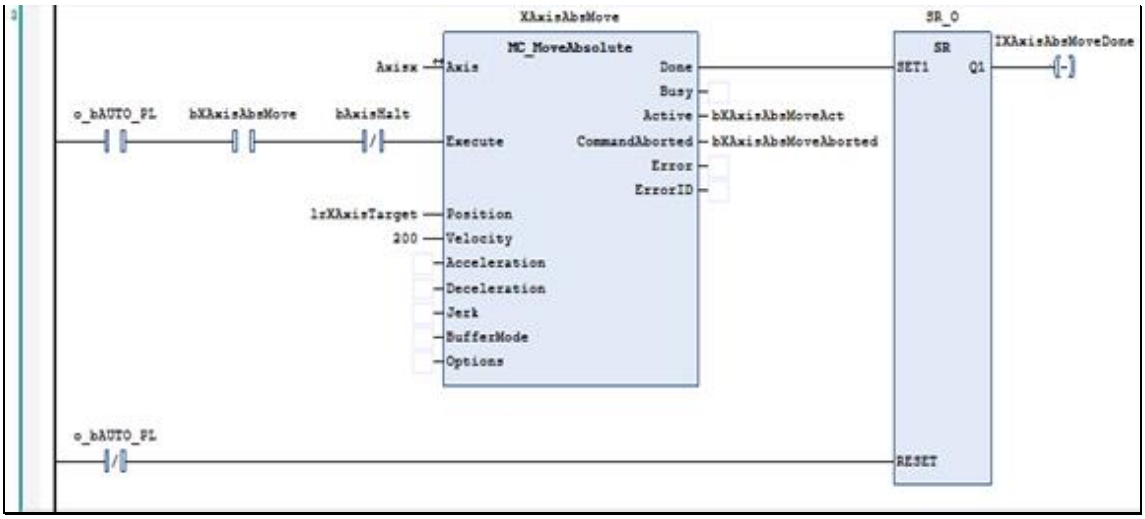
Käyttöliittymän “Calibration/Homing”-painike on kytketty tulon muuttujaan ”bXAxisToHome”. Kotoutus-toiminto ”nollaa” etäisyyttä lukevan enkooderin vertailu-arvon.




Kuva 61. Käytön akselin kotouttaminen ”MC_Home”-funktioblokilla.

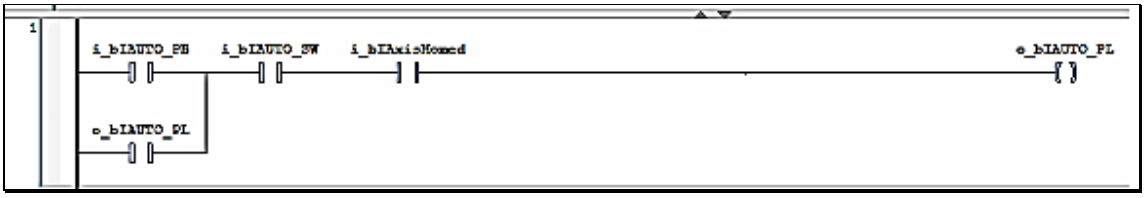
### 6.2.4 Käytön automaattinen paikka-ohjattu operointi

Käyttöliittymän kiertokytkin on liitetty tulopuolen muuttujaan ”o_bAUTO_PL” ja painonappi ”X-Axis Move” muuttujaan ”bXAxisAbsMove”. Paikan arvo kirjoitetaan ”lrXaxisTarget”-muuttujaan.[13]




Kuva 62. Paikka-ohjatun liiketilan toteutus, jossa käytetään ”MC_MoveAbsolute”-funktioblokkia.[13]

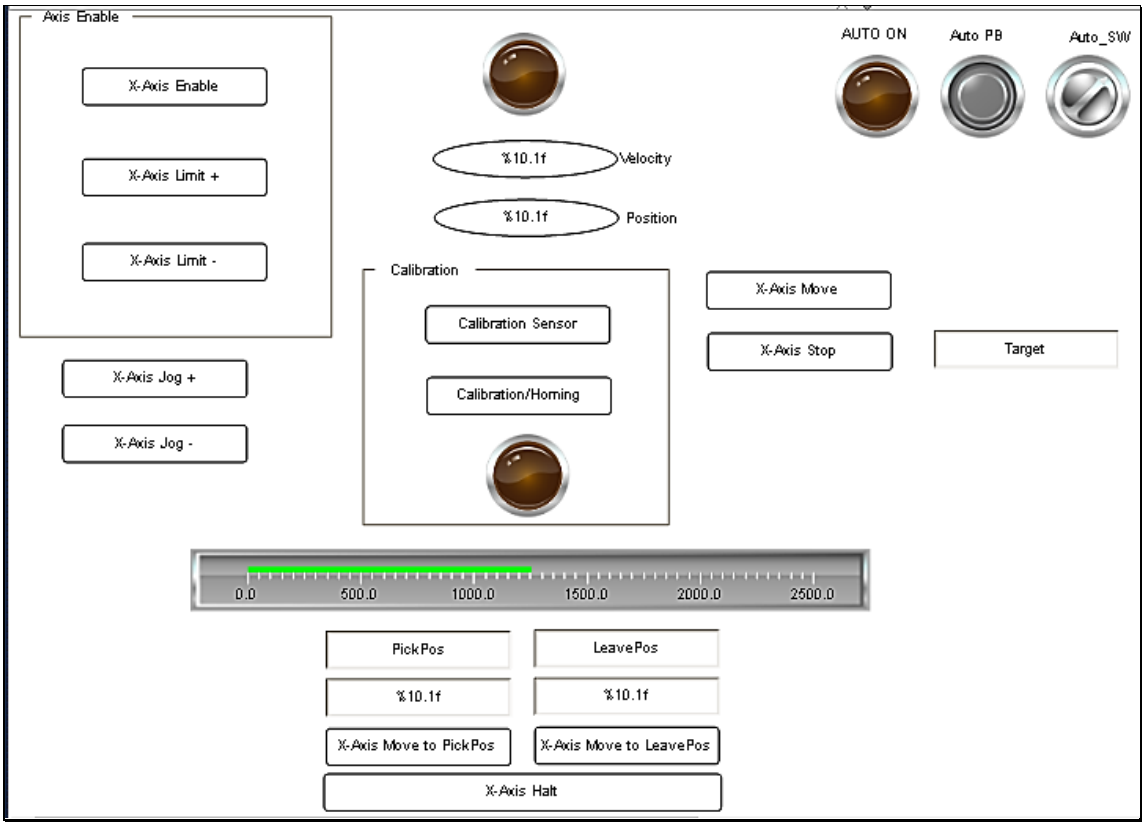
Kuvassa 63. on paikka-ohjatun liiketilan aloituksen ohjauslogiikka, jonka lähtöarvon muuttuja ”o_bIAUTO_PL” tila ”TOSI” on ensimmäisenä ehtona kuvan 62 absoluuttisen liiketilan funktioblokin suorituksessa. Ohjauslogiikan tulomuuttujat ”i_bIAUTO_PB”, ”i_bIAUTO_SW” ja ”i_bIAxisHomed” on linkitetty käyttöliittymän paikka-ohjauksen ”Auto_PB”-painonappiin ja ”Auto_SW”-kiertokytkimeen sekä kotoutuksen ”Calibration/Homing”-painonappiin.




Kuva 63. Paikka-ohjatun liiketilan aloituksen ohjauslogiikka.

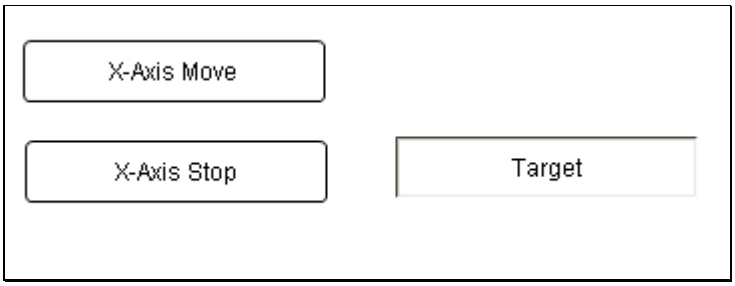
## 6.3 PLC-ohjelman paikallinen käyttöliittymä

Kuvassa 64. on servokäytön paikallinen käyttöliittymä.




Kuva 64. Servokäytön paikallinen käyttöliittymä.

Kuvassa 65. on paikka-ohjatun, yhden paikka muuttujan käyttöliittymän osio. ”Target”-kenttään annetaan halutun paikan arvo.



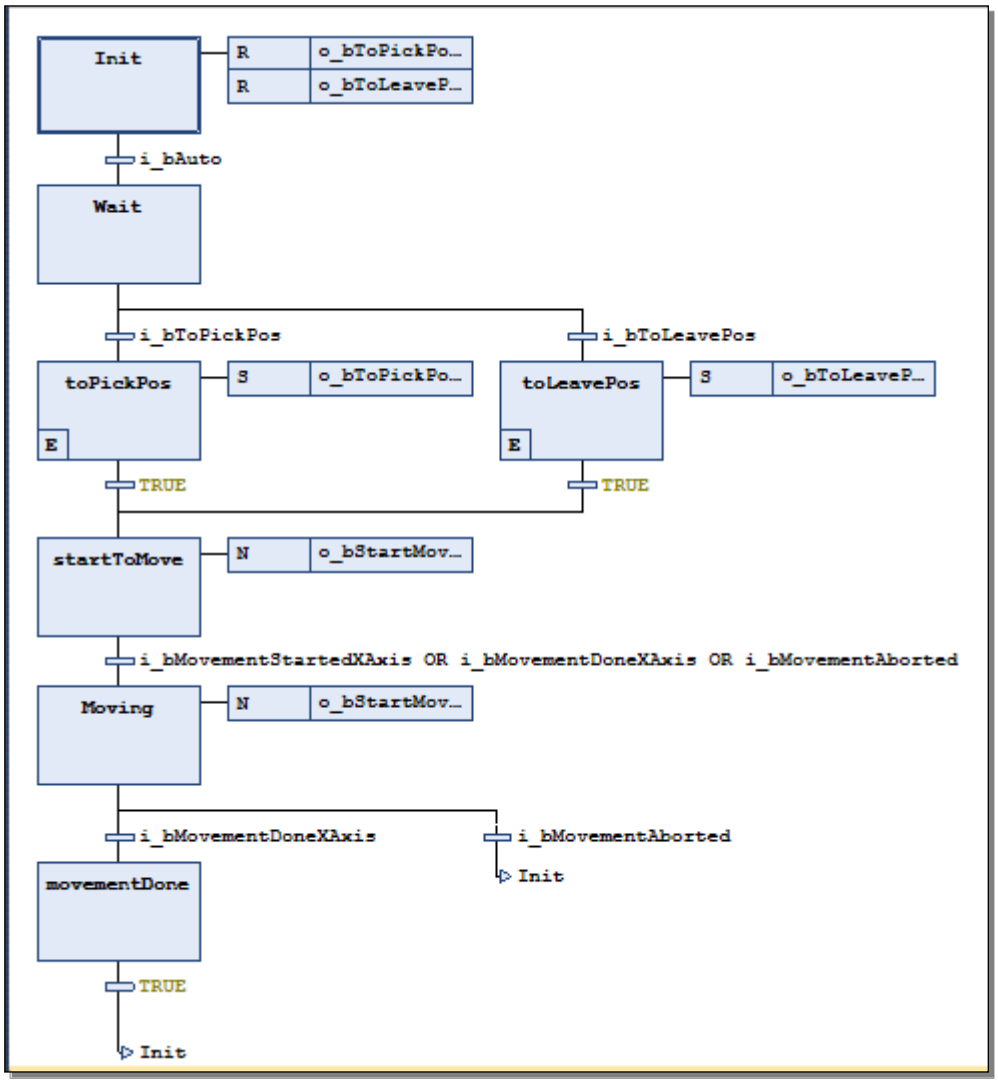

Kuva 65. Paikka-ohjattu käyttöliittymä yhdelle etäisyyden arvolle.

Kuvan 66. paikka-ohjattuun, kahden paikka-arvon käyttöliittymään voidaan antaa etäisyydelle alkuarvo (”PickPos”) ja loppuarvo (”LeavePos”).


Kuva 66. Kuvassa on paikka-ohjattu käyttöliittymä kahdelle etäisyyden arvolle. Samaan muuttujaan talletetaan alku- ja loppuarvo ohjauslogiikan tilakoneen mukaisesti.

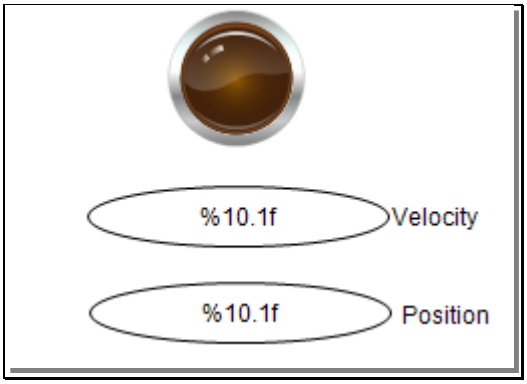
Kuvan 66. kahden paikan arvon käyttöliittymää ohjataan kuvan 67. mukaisella SFC-kielen ohjaus- ja valintalogiikalla. Kuvan 67. logiikka on tilakone, jossa kuvataan tilat ja niihin liittyvät operaatiot, sekä ehdot seuraavaan tilaan siirtymiseksi. Paikan arvot syötetään käyttöliittymän alkuarvon (”PickPos”) ja loppuarvon (”LeavePos”) kenttiin, jossa

ne tallennetaan muuttujiin ” i_lrXAxisPickPos” ja ” i_lrXAxisLeavePos”. Nämä parametrit sijoitetaan suoritusjärjestyksessä ensin alku- ja sitten loppuarvo kohdemuuttujaan ” o_lrXAxisTargetPos”. Pääohjelmassa (”Main”) tämä kyseinen kohdemuuttuja sijoitetaan funktioblokki ”MC_MoveAbsolute”:n tulopuolen muuttujaan ”lrXaxisTarget”. [17]




Kuva 67. Käyttöliittymän kahden paikkamuuttujan arvon SFC-kielinen ohjaus- ja valintalogiikka.[17]

Kuvassa 68. on nähtävissä nopeuden (”velocity”) ja paikan (”position”) näyttö.




Kuva 68. Käytön nopeuden ja paikan ilmaisevat näyttökentät.

# 7 Yhteenveto

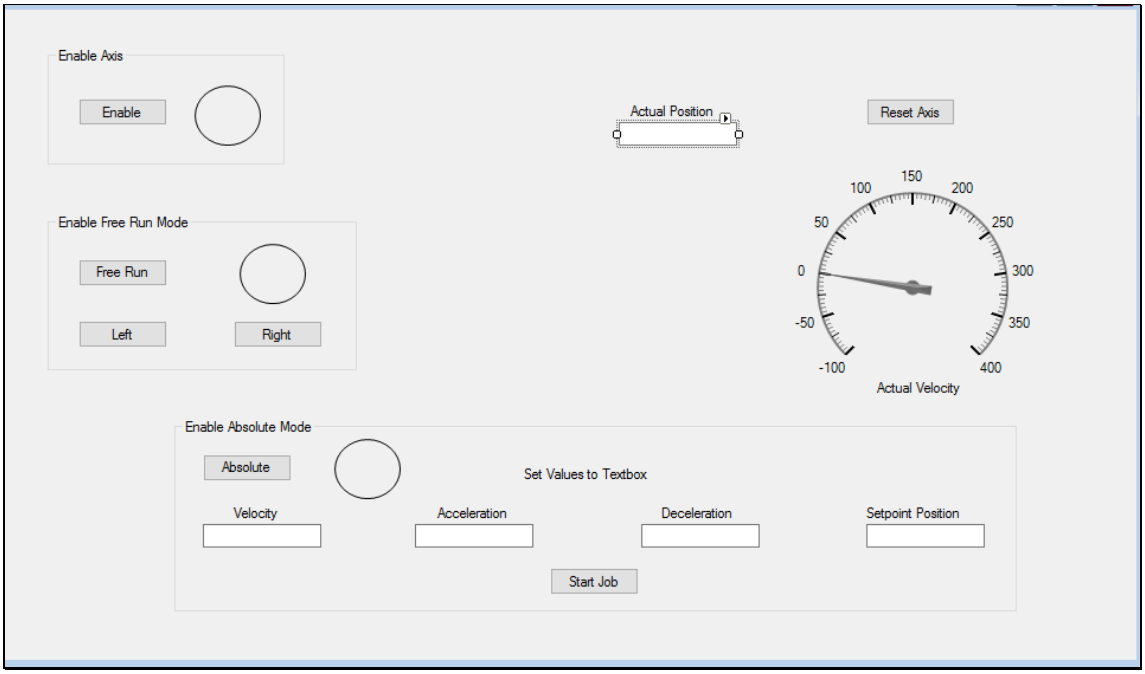
Työn päätavoite saavutettiin. Servomoottorikäytön rakentaminen Twincat 3-ympäristöön, jossa on paikallinen käyttöliittymä. Käyttöliittymä kykenee näyttämään servomoottorikäytön prosessin muuttujia, nopeutta ja paikkaa.

Etäkäyttöliittymän toteutus jäi kokeelliseksi ja keskeneräiseksi. Tämä johtui osittain ajan ja resurssien puutteesta. Työtä tehtiin Twincat 3 (XAE) kehitysympäristön .NET-kielistä C#-kielellä. Tämä siksi, että lähdemateriaalina käytetyt Beckhoffin ohjelmointi esimerkit olivat pääosin C#:lla tehtyjä. ”.NET” -kielistä toinen vaihtoehto olisi ollut Visual Basic.

## 7.1 Etäkäyttöliittymäsovelluksen kehityksen vaiheita

Etäkäyttöliittymä on PC:n sovellusohjelma. Se on palvelinohjelma, joka kommunikoi PLC-ohjelman kanssa, lukien ja kirjoittaen PLC:n muuttujia ja ohjaten sen eri toimintoja. Tässä työssä hyödynnettiin aiemmin tutkittua TwinCAT-ADS rajapintaa ja sen TwinCAT ADS.dll-kirjaston tarjoamia metodeja.

Käyttöliittymän (”GUI-Graphical User Interface”) graafinen ulkoasu oli kuvan 69. kaltainen.




Kuva 59. Etäkäyttöliittymän ulkoasu.

Kuvan 69. mukaisesti käyttöliittymän ohjaustoiminnot toteutettiin painokytkimillä. Parametrien luku ja kirjoitus kenttinä käytetään ”textbox” tyyppisiä graafisia olioita. Näiden lisäksi hetkellistä nopeutta (”Actual Velocity”) lukevana näyttönä käytettiin ”Agauge.dll” nimistä C#-aliohjelmaa.

Ohjelman kulku käynnistyy käyttöliittymän vasemmasta yläkulmasta käytön ”akselin” aktivoimisella painonapilla ”Enable”. Tämä painonappi kirjoittaa PLC-ohjelmassa ”Glo-

bal” -muuttujissa määriteltyä binääristä (BOOL) muuttujaa ”bEnable”. ”bEnable”-muuttujan aktivoituessa arvoon ”TOSI”, käynnistyy funktioblokki ”MC_POWER”, joka mahdollistaa moottorin pyörimiskäskyt ja moottorin pyörimisen molempiin suuntiin.

### 7.1.1 Muuttujakahvojen luonti PLC-muuttujien perusteella

Jokaiselle PLC symbolimuuttujalle luotiin oma kahvamuuttujansa C#-ohjelmassa.

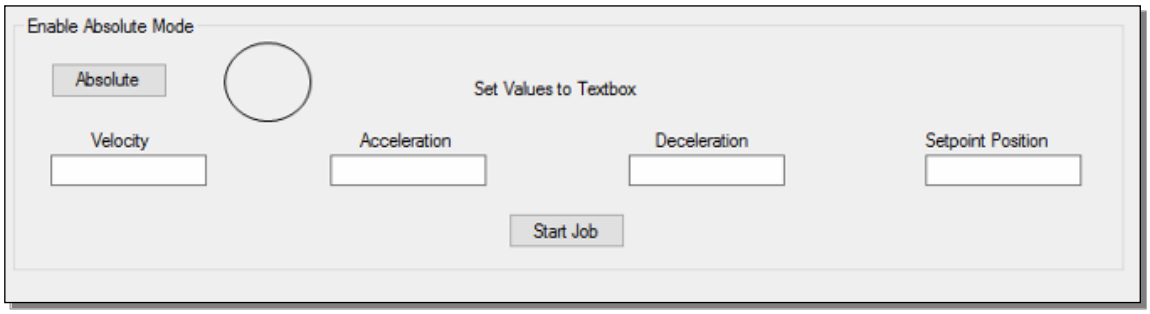
Metodien kutsu symbolinimen perusteella toteutettiin kuvan 70.mukaisesti. Muuttujakahvat luotiin ”tcClient.CreateVariableHandle”-metodilla, jossa PLC-symbolimuuttujan nimi ja hakemistopolku kirjoitetaan parametriksi.

```
try
{
    tcClient.Connect(851);
    tcClient.AdsNotificationEx += new AdsNotificationExEventHandler(tcClient_AdsNotificationEx);
    //btnDeleteNotifications.Enabled = false;

    hbEnableAxis = tcClient.CreateVariableHandle("MAIN.GVL.bEnable");
    hbFreeRun = tcClient.CreateVariableHandle("MAIN.GVL.bFreeRun");
    hbMoveLeft = tcClient.CreateVariableHandle("MAIN.GVL.bMoveLeft");
    hbMoveRight = tcClient.CreateVariableHandle("MAIN.GVL.bMoveRight");
    hbEnableAbsolute = tcClient.CreateVariableHandle("MAIN.GVL.bEnableAbsolute");
    hbMoveAbsolut = tcClient.CreateVariableHandle("MAIN.GVL.bMoveAbsolut");
    hbStop = tcClient.CreateVariableHandle("MAIN.GVL.bStop");
    hbReset = tcClient.CreateVariableHandle("MAIN.GVL.bReset");

    hVar = tcClient.CreateVariableHandle("MAIN.PLCVar");
    //hlrealvelo = adsClient.CreateVariableHandle("MAIN.lrVelocity_Move_Ab_Axis_1");
    //hlrealaccele = adsClient.CreateVariableHandle("MAIN.lrAcc_Axis_1");
    //hlrealdecele = adsClient.CreateVariableHandle("MAIN.lrDecel_Axis_1");
    //hlrealspposi = adsClient.CreateVariableHandle("MAIN.lrPosition_Drive_to");
    hlrealactvelo = tcClient.CreateVariableHandle("MAIN.lrActualVelocity");
    hlrealactposi = tcClient.CreateVariableHandle("MAIN.lrActualPosition");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Kuva 60. Muuttujakahvojen luonti PLC-symbolimuuttujille.

### 7.1.2 Eräitä käyttöliittymäsovelluksen tapahtumakäsittelijöitä

Kuvassa 71. on näkymä paikka-ohjatun, eli absoluuttisen liiketilan osuuteen käyttöliittymästä. Tilan aktivointi tapahtuu ”Absolute”-painonapista, jonka jälkeen tarkoituksena on syöttää ao. tekstikenttiin parametrit muuttujille nopeus (”Velocity”), kiihtyvyys (”Acceleration”), hidastuvuus (”Deceleration”) ja paikan arvo (”Setpoint Position”). Tämän jälkeen tekstikenttä objekteihin luetut arvot kirjoitetaan PLC-ohjelmaan painonapilla (”Start Job”).




Kuva 71. Käyttöliittymäsovelluksen paikka-ohjatun liiketilan osuus.

Muista käyttöliittymän objekteista poiketen ”absolute mode”-tilan ”textbox”-tekstikenttien arvot luetaan ”struct”-tyyppisinä tietorakenteina ”Start Job”-napin painalluksena.

Kuvassa 72. on tapahtumankäsittelijän tutkielma edellisen kuvan käyttöliittymän tekstikenttien kirjoittamiseksi “AdsStream” luokkaan ”BinaryWriter” luokan metodilla.

```
private void btnWrite_Click(object sender, EventArgs e)
{
    AdsStream dataStream = new AdsStream(19);
    BinaryWriter binWrite = new BinaryWriter(dataStream);

    dataStream.Position = 0;
    try
    {
        //Fill stream according to the order with data from the text boxes
        binWrite.Write(float.Parse(tbVelocity.Text));
        binWrite.Write(float.Parse(tbAcceleration.Text));
        binWrite.Write(float.Parse(tbDeceleration.Text));
        binWrite.Write(float.Parse(tbSPPosition.Text));

        //Write complete stream in the PLC
        tcClient.Write(hVar, dataStream);
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message);
    }
}
```

Kuva 72. Tapahtumakäsittelijä arvokenttien parametrien kirjoittamiseen ”AdsStream”-luokkaan.

### 7.1.3 Loppusanat

Etäkäyttöliittymän ohjelmointiosuus oli kokeiluluontoinen, ylimääräinen osa. Se oli laajuudeltaan ja vaativuudeltaan tätä päättötyötä suurempi, mutta puolusti paikkaansa henkilökohtaisena opintoprojektina .NET C#-ohjelmointiin.

## Lähteet


1 Beckhoff Automation PDF-dokumentti osoitteessa:https://download.beckhoff.com/download/document/Application_Notes/DK9322-0110-0024.pdf

2 Beckhoff Automation PDF-dokumentti osoitteessa: https://download.beckhoff.com/download/document/Application_Notes/DK9322-0413-0070.pdf

3 www-luento (”TwinCAT ADS Beckhoff protocol as the connecting link for TwinCAT modules. Beckhoff”) osoitteessa:http://multimedia.beckhoff.com/webinar/en/webinar-twincat-ads/default.htm,

4 www-dokumentti osoitteessa:https://infosys.beckhoff.com/english.php?content=../content/1033/tcsample_net/html/tcsample_net_intro.htm&id=

5 PDF-dokumentti www-osoitteessa:https://download.beckhoff.com/download/Document/io/ethercat-terminals/el72x1-0010en.pdf

6 www-dokumentti.https://www.scribd.com/doc/121408255/Module-4-Introduction-to-ADS-Programming

7 www-dokumentti osoittees-sa:https://www.scribd.com/document/331156353/BECKHOFF-TC3-002-TwinCAT-3-2012

8 www-dokumentti osoitteessa:. http://www.contactandcoil.com/twincat-3-tutorial/introduction-to-motion-control/

9 PDF-dokumentti www-osoitteessa: https://www.scribd.com/document/294132015/TC3-E-04

10 PDF-dokumentti www-osoitteessa: http://www.infoplc.net/files/descargas/beckhoff/infoplc_net_guide_for_twincat_nc_control.pdf

11 www-dokumentti osoittees-sa:.https://infosys.beckhoff.com/english.php?content=../content/1033/bc9000/html/bt_bx9000adsprotocol.htm&id=

12 Youtube-video (“Control XAxis from HMI”) www-osoitteessa: https://www.youtube.com/watch?v=qN3V-hoO9TE, SAMKAutomation Youtube-tili.

13 Youtube-video (“How to setup automatic control and make absolute movement”) www-osoitteessa: https://www.youtube.com/watch?v=Yw6UFJYLeVE, ,SAMKAutomation Youtube-tili.

14 Youtube-video (“How to setup jog control”) www-osoitteessa https://www.youtube.com/watch?v=j1fzU9l0H-s, SAMKAutomation Youtube-tili.

15 Youtube-video (“How to setup PLC and Visualization”) www-osoitteessa https://www.youtube.com/watch?v=0HqF1m10q0Y, SAMKAutomation Youtube-tili.

16 Youtube-video (“How to setup calibration”) www-osoitteessa: https://www.youtube.com/watch?v=Lq_2vX2S_SM&t=306s, SAMKAutomation Youtube-tili.

17 Youtube-video(“How to setup sequence”) www-osoitteessa:.https://www.youtube.com/watch?v=LDeSxK1uWnQ, SAMKAutomation Youtube-tili.

18 PDF-dokumentti www-osoitteessa:http://www.plcopen.org/pages/tc2_motion_control/downloads/flyer_creating_reusable.pdf

PLC-ohjelman listaus

```
VAR_GLOBAL//
        bXAxisEnable :      BOOL;
        bXAxisJogPos :      BOOL;
        bXAxisJogNeg :      BOOL;
        bXAxisToHome :      BOOL;
        bXAxisCalibrated    :       BOOL;
        bXAxisToHomeAct :   BOOL;
```

```
        bXAxisMoveHMI :         BOOL;
        bXAxisMoveToPickPos     :       BOOL;
        bXAxisMoveToLeavePos    :       BOOL;
        bXAxisAbsMove           :       BOOL;
        bXAxisAbsMoveAct :      BOOL;
        bXAxisAbsMoveDone :     BOOL;
        bXAxisAbsMoveAborted :  BOOL;
        bXAxisHalt    :         BOOL;
        bXAxisStop : BOOL;
        lrXLeavePos :           LREAL;
        lrXPickPos    :         LREAL;
        lrXAxisTarget :         LREAL;
END_VAR


VAR_GLOBAL//IO
        I_bXAxisLimitSWPos      AT      %I*     :       BOOL;
        I_bXAxisLimitSWNeg      AT      %I*     :       BOOL;
        I_bXAxisCalibSw         AT      %I*     :       BOOL;
        I_bAUTO_SW AT %I*       :BOOL;
        I_bAUTO_PB AT %I*       : BOOL;
        O_bAUTO_PL AT %Q*       : BOOL;
```

```
END_VAR


VAR_GLOBAL//NC
        Axisx:AXIS_REF;
        AxisxStatus:MC_ReadStatus;
END_VAR


PROGRAM MAIN
VAR
        ReadXAxisStatus: MC_ReadStatus;
        Init_seq :      BOOL;
        //MoveXAxis :           POU_XAxisMove;
        fbCBControl : CB_FB;


        Axisx: AXIS_REF;
        bXAxisEnable: BOOL;
        AxisxStatus: ST_AxisStatus;
        lrTargetXAxis: LREAL;
        SFCInit: INT;
        i_IAUTO: BOOL;
        i_ICalibrated: BOOL;
```

```
        i_IMoveXAxis: INT;
        IXAxisMoveHMI: BOOL;


        IXAxisAbsMoveDone: BOOL;
        IXAxisAbsMove: BOOL;
        fbAxisCtrl      :       XAxisCtrl;
        IXAxisAbsMoveAct: BOOL;
        i_bAuto: BOOL;
        o_lrXAxisTargetPos: LREAL;
        i_bMovementStartedXAxis: BOOL;
        fbCBContol: CB_FB;
        //AUTO_PL: BOOL;
        I_bAUTO_PL: BOOL;
END_VAR


ReadXAxisStatus(
        Axis:= Axisx,
        Enable:= bXAxisEnable,
        Valid=> ,
        Busy=> ,
        Error=> ,
```

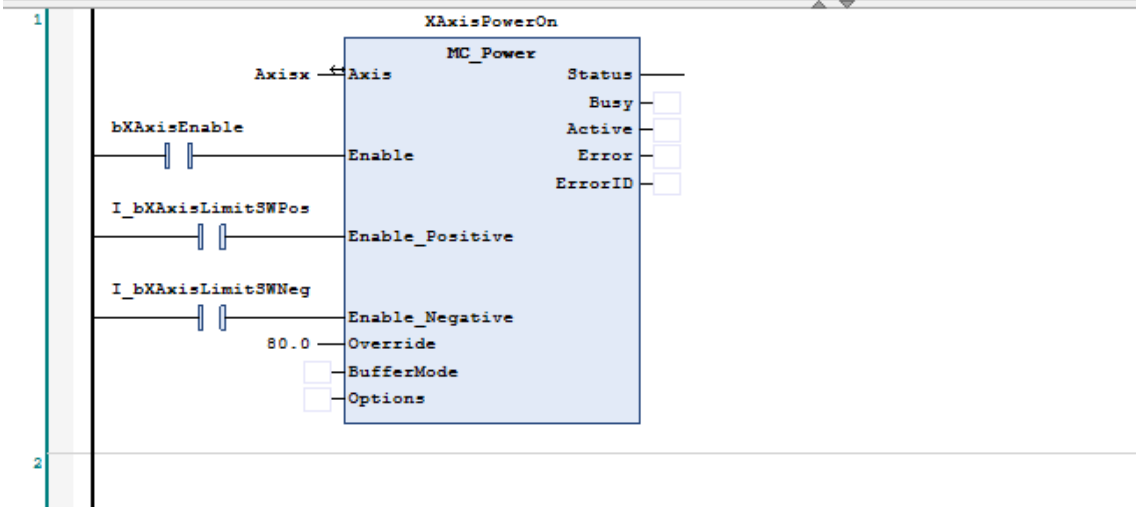
```
                ErrorID=> ,

                ErrorStop=> ,

                Disabled=> ,

                Stopping=> ,

                StandStill=> ,

                DiscreteMotion=> ,

                ContinuousMotion=> ,

                SynchronizedMotion=> ,

                Homing=> ,

                ConstantVelocity=> ,

                Accelerating=> ,

                Decelerating=> ,

                Status=> AxisxStatus);



                POU_NCPowerON();

                POU_XAxisCtrl();



                lrTargetXAxis:=300;

fbCBCtol(i_bIAUTO_PB:=I_bAUTO_PB,i_bIAUTO_SW:=I_bAUTO_SW,i_bIAxisHomed
:=AxisxStatus.Homed,o_bIAUTO_PL=>O_bAUTO_PL);



//              MoveXAxis(
```

```
//                          SFCInit:= NOT AUTO_PL,
                //          i_IAUTO:= AUTO_PL,
                            //i_ICalibrated:= AxisxStatus.Homed,
                            //i_IMoveXAxis:= IXAxisMoveHMI,
                            //i_IMovementStarted:=IXAxisAbsMoveAct,
                            //7i_IMovementExecuted:=IXAxisAbsMoveDone,
                            //o_IStartMovement=>IXAxisAbsMove,
                            //o_lrTargetPos=>lrTargetXAxis);//

                fbAxisCtrl (
                            SFCInit:=NOT O_bAUTO_PL         ,
                            i_bAuto:=O_bAUTO_PL    ,
    i_bToPickPos:=bXAxisMoveToPickPos    ,
    i_bToLeavePos:=bXAxisMoveToLeavePos    ,
    i_bMovementStartedXAxis:=bXAxisAbsMoveAct    ,
    i_bMovementDoneXAxis:=bXAxisAbsMoveDone    ,
    i_bMovementAborted:=bXAxisAbsMoveAborted AND NOT bXAxisHalt         ,
    i_lrXAxisPickPos:=lrXPickPos ,
    i_lrXAxisLeavePos:=lrXLeavePos,
                o_bStartMovementXAxis=>bXAxisAbsMove,
    o_bToPickPosAct=>bXAxisMoveToPickPos ,
```

o_bToLeavePosAct=>bXAxisMoveToLeavePos ,

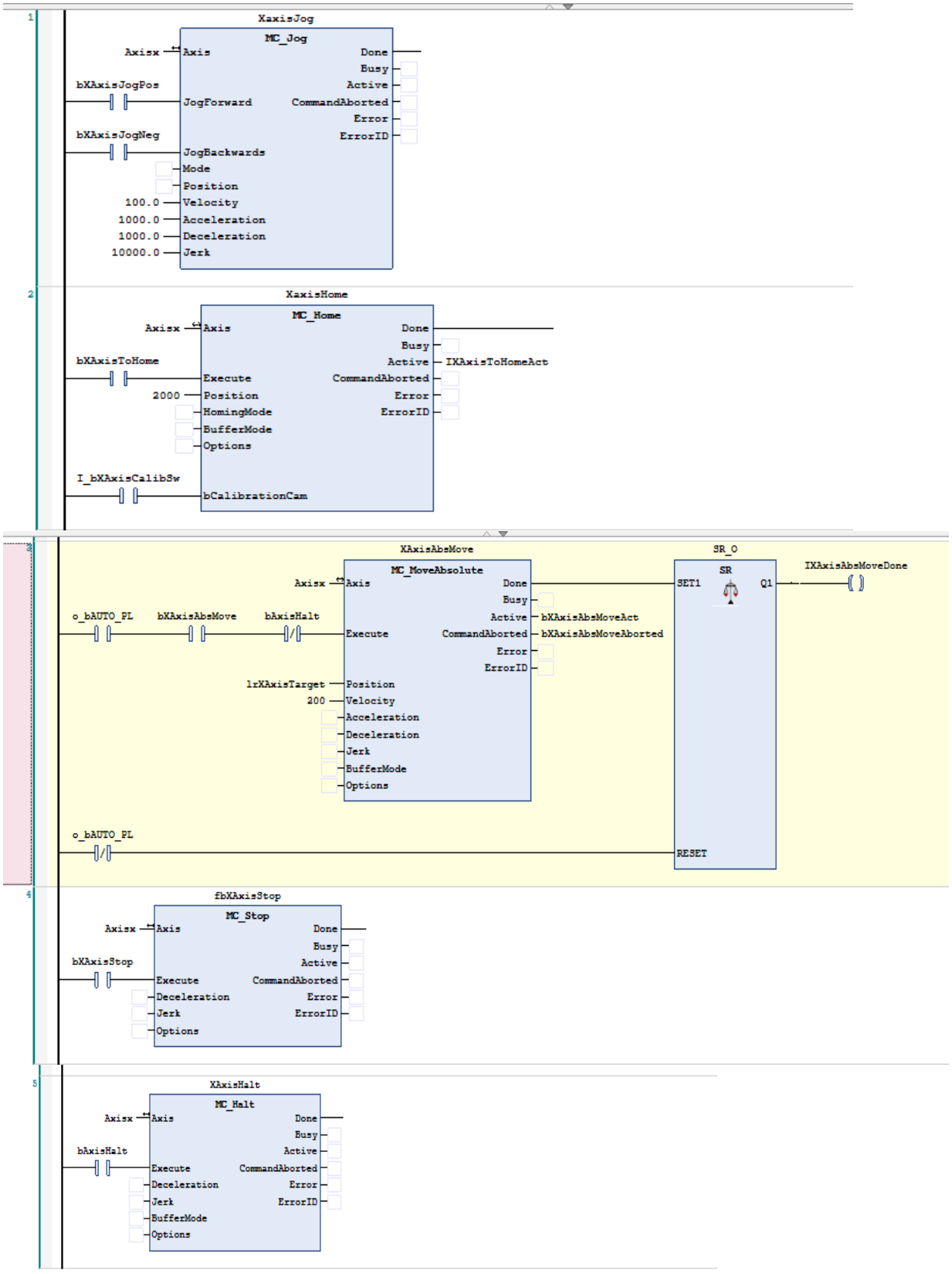
o_lrXAxisTargetPos=>lrXAxisTarget);

```
PROGRAM POU_NCPowerOn
VAR

    XAxisPowerOn: MC_Power;
    bXAxisEnable: BOOL;
END_VAR
```



```
FUNCTION_BLOCK CB_FB
VAR_INPUT
    i_bIAUTO_SW :   BOOL;
    i_bIAUTO_PB :   BOOL;
    i_bIAxisHomed   :   BOOL;
END_VAR
VAR_OUTPUT
    o_bIAUTO_PL :   BOOL;
END_VAR
```


```
PROGRAM POU_XAxisCtrl
VAR
    XaxisJog : MC_Jog;
    XaxisHome : MC_Home;
    XAxisAbsMove: MC_MoveAbsolute;

    IXAxisMoveHMI: BOOL;
    IAxisAbsMoveAct: BOOL;
    SR_0: SR;
    IXAxisAbsMoveDone: BOOL;
    MC_Halt_0: MC_Halt;
    XAxisHalt: MC_Halt;
    IXAxisStop: BOOL;
    IXAxisJogPos: BOOL;
    IXAxisJogNeg: BOOL;
    IXAxisToHome: BOOL;
    XAxisCalibSw: BOOL;
    IXAxisToHomeAct: BOOL;
    IXAxisAbsMoveAct: BOOL;
    Axisx: AXIS_REF;
    fbXAxisStop: MC_Stop;
    bAxisHalt: BOOL;
END_VAR
```


1
XaxisJog
MC_Jog
Axisx
Axis
Done
Busy
Active
bXAxisJogPos
JogForward
CommandAborted
Error
bXAxisJogNeg
ErrorID
JogBackwards
Mode
Position
100.0
Velocity
1000.0
Acceleration
1000.0
Deceleration
10000.0
Jerk
2
XaxisHome
MC_Home
Axisx
Axis
Done
Busy
bXAxisToHome
Active
IXAxisToHomeAct
Execute
CommandAborted
2000
Position
Error
HomingMode
ErrorID
BufferMode
Options
I_bXAxisCalibSw
bCalibrationCam
3
XAxisAbsMove
MC_MoveAbsolute
SR_0
SR
IXAxisAbsMoveDone
Axisx
Axis
Done
SET1
Q1
Busy
o_bAUTO_PL
bXAxisAbsMove
bAxisHalt
Active
bXAxisAbsMoveAct
Execute
CommandAborted
bXAxisAbsMoveAborted
Error
ErrorID
lrXAxisTarget
Position
200
Velocity
Acceleration
Deceleration
Jerk
BufferMode
Options
o_bAUTO_PL
RESET
4
fbXAxisStop
MC_Stop
Axisx
Axis
Done
Busy
bXAxisStop
Active
Execute
CommandAborted
Deceleration
Error
Jerk
ErrorID
Options
5
XAxisHalt
MC_Halt
Axisx
Axis
Done
Busy
bAxisHalt
Active
Execute
CommandAborted
Deceleration
Error
Jerk
ErrorID
BufferMode
Options

```
1  FUNCTION_BLOCK XAxisCtrl
2  VAR_INPUT
3      SFCInit:BOOL;
4      i_bAuto     :     BOOL;
5      i_bToPickPos :     BOOL;
6      i_bToLeavePos     :     BOOL;
7      i_bMovementStartedXAxis     : BOOL;
8      i_bMovementDoneXAxis     : BOOL;
9      i_bMovementAborted     : BOOL;
10     i_lrXAxisPickPos     : LREAL;
11     i_lrXAxisLeavePos     : LREAL;
12
13 END_VAR
```